Motorbike Statistics

Generated by Doxygen 1.8.13

Contents

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:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

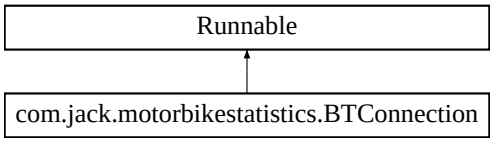
Chapter 5

Class Documentation

5.1 com.jack.motorbikestatistics.BTConnection Class Reference

Thread class for a new bluetooth connection to a device.

Inheritance diagram for com.jack.motorbikestatistics.BTConnection:



Public Member Functions

- BTConnection (BluetoothDevice btDevice) throws IOException

  Constructor for BTConnection class.
- void setRXHandler (Handler newHandler)

  Setter 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()

```
com.jack.motorbikestatistics.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

btDevice	- Device used for creating connection.

References com.jack.motorbikestatistics.BTConnection.btDevice, and com.jack.motorbikestatistics.BTConnection.↩ connect().

```
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 com.jack.motorbikestatistics.BTConnection.setRXHandler (
              Handler newHandler )  [inline]
```

Setter function for RXHandler.

Parameters

newHandler	- The new Handler where RX'd data will be sent to.

```
108                                                 {
109         RXHandler = newHandler;
110     }
```

5.1.3.2 run()

```
void com.jack.motorbikestatistics.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).

References com.jack.motorbikestatistics.BTConnection.close(), com.jack.motorbikestatistics.BTConnection.is↩ Connected(), and com.jack.motorbikestatistics.BTConnection.isRunning().

```
121                     {
122         InputStream RXStream;
123
124         /* Indicate that we are now running main thread */
125         running = true;
126
127         if (isConnected()) {
128             /* Get our input stream for receiving bytes */
129             try {
130                 RXStream = btSocket.getInputStream();
131             } catch (IOException e) {
```

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

5.1.3.3 isRunning()

```
boolean com.jack.motorbikestatistics.BTConnection.isRunning ( )  [inline]
```

Function to check whether main connection thread is running.

Returns

boolean - Whether thread is running.

References com.jack.motorbikestatistics.BTConnection.running.

```
208                                 {
209         return running;
210     }
```

5.1.3.4 isConnected()

```
boolean com.jack.motorbikestatistics.BTConnection.isConnected ( )  [inline]
```

Function to check whether BT connection is still valid.

Returns

boolean - Whether connection is still available.

```
216                                   {
217         boolean result = false;
218
219         if (btSocket != null) {
220             if (btSocket.isConnected())
221                 result = true;
222         }
223         return result;
224     }
```

5.1.3.5 connect()

```
void com.jack.motorbikestatistics.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.

References com.jack.motorbikestatistics.BTConnection.close().

```
232                                             {
233
234         /* Attempt to make connection to remote device, throw exception if not */
235         try {
236             btSocket = btDevice.createRfcommSocketToServiceRecord(
      uuid);
237         } catch (IOException e) {
238             Log.e(TAG, "Unable to create RFCOMM", e);
239             throw e;
240         }
241
242         try {
243             btSocket.connect();
244         } catch (IOException e) {
245             Log.e(TAG, "Unable to connect", e);
246
247             /* Close our socket as unable to connect */
248             try {
249                 this.close();
250             } catch (IOException e2) {
251                 throw e2;
252             }
253             throw e;
254         }
255     }
```

5.1.4 Member Data Documentation

5.1.4.1 txHandler

```
final Handler com.jack.motorbikestatistics.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 com.jack.motorbikestatistics.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()

```
com.jack.motorbikestatistics.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

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

References com.jack.motorbikestatistics.BTDeviceItem.device, com.jack.motorbikestatistics.BTDeviceItem.iconID, and com.jack.motorbikestatistics.BTDeviceItem.status.

```
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 com.jack.motorbikestatistics.BTDeviceItem.getConnection ( )  [inline]
```

Getter for the bluetooth connection of specified device.

Returns

BTConnection - Connection between app & logging device.

References com.jack.motorbikestatistics.BTDeviceItem.connection.

```
33                                      {
34          return connection;
35      }
```

5.2.3.2 setConnection()

```
void com.jack.motorbikestatistics.BTDeviceItem.setConnection (
              BTConnection newConn )  [inline]
```

Setter for setting the DeviceItem object's connection.

Parameters

newConn	- New connection between app & logging device.

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

5.2.3.3 getDevice()

```
BluetoothDevice com.jack.motorbikestatistics.BTDeviceItem.getDevice ( )  [inline]
```

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

Returns

BluetoothDevice - The bluetooth device object.

References com.jack.motorbikestatistics.BTDeviceItem.device.

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

5.2.3.4 getStatus()

```
String com.jack.motorbikestatistics.BTDeviceItem.getStatus ( )  [inline]
```

Getter for current status of BTDeviceItem.

Returns

String - Current status: unpaired, paired or connected.

References com.jack.motorbikestatistics.BTDeviceItem.status.

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

5.2.3.5 setStatus()

```
void com.jack.motorbikestatistics.BTDeviceItem.setStatus (
              String newStatus )  [inline]
```

Setter for current status of BTDeviceItem.

Parameters

newStatus	- New string for status.

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

5.2.3.6 getIconID()

```
int com.jack.motorbikestatistics.BTDeviceItem.getIconID ( )  [inline]
```

Getter for icon ID to use in ListView.

Returns

int - Icon ID to use.

References com.jack.motorbikestatistics.BTDeviceItem.iconID.

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

5.2.3.7 setIconID()

```
void com.jack.motorbikestatistics.BTDeviceItem.setIconID (
            int newID )  [inline]
```

Setter for icon ID to use in ListView.

Parameters

newID	- New icon ID to use.

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

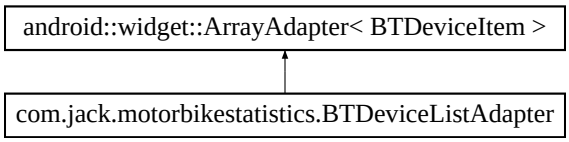
The documentation for this class was generated from the following file:

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

5.3 com.jack.motorbikestatistics.BTDeviceListAdapter Class Reference

Adapter class used for displaying bluetooth devices.

Inheritance diagram for com.jack.motorbikestatistics.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()

```
com.jack.motorbikestatistics.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

cnt	- Context of the adapter to be operating in.
layout↩ ResourceId	- Resource ID for current layout.
data	- ArrayList of devices to display in ListView.

References com.jack.motorbikestatistics.BTDeviceListAdapter.data, and com.jack.motorbikestatistics.BTDevice↩ ListAdapter.layoutResourceId.

```
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 com.jack.motorbikestatistics.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

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

Returns

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

References com.jack.motorbikestatistics.BTDeviceItem.getDevice(), com.jack.motorbikestatistics.BTDeviceItem.↩ getIconID(), and com.jack.motorbikestatistics.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 com.jack.motorbikestatistics.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 chcek whether A is greater than B.
- boolean lessThan (Number a, Number b)

  Function to chcek 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 = 0.0

  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.

5.4.1 Detailed Description

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

5.4.2 Constructor & Destructor Documentation

5.4.2.1 DataItem() [1/2]

```
com.jack.motorbikestatistics.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

name	- Name of the data item.
avgMinMax	- Whether additive functionality shall be available.

References com.jack.motorbikestatistics.DataItem< T >.name.

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

5.4.2.2 DataItem() [2/2]

```
com.jack.motorbikestatistics.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

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

References com.jack.motorbikestatistics.DataItem< T >.name.

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

5.4.3 Member Function Documentation

5.4.3.1 getName()

`String com.jack.motorbikestatistics.DataItem< T >.getName ( )  [inline]`

Getter for name of data item.

Returns

String - DataItem name.

References com.jack.motorbikestatistics.DataItem< T >.name.

```
82                              {
83          return name;
84      }
```

5.4.3.2 getEnabledAvgMinMax()

`boolean com.jack.motorbikestatistics.DataItem< T >.getEnabledAvgMinMax ( )  [inline]`

Getter for whether additional functionality enabled.

Returns

boolean - Averaging, Minimum & Maximum enabled.

References com.jack.motorbikestatistics.DataItem< T >.enableAvgMinMax.

```
90                                      {
91          return enableAvgMinMax;
92      }
```

5.4.3.3 getCurrent()

```
T com.jack.motorbikestatistics.DataItem< T >.getCurrent ( )  [inline]
```

Getter for current reading value.

Returns

T - Current reading value.

References com.jack.motorbikestatistics.DataItem< T >.current.

```
98                          {
99          return current;
100     }
```

5.4.3.4 getAverage()

```
Double com.jack.motorbikestatistics.DataItem< T >.getAverage ( )  [inline]
```

Getter for average of readings.

Returns

Double - Average of all readings.

References com.jack.motorbikestatistics.DataItem< T >.average.

```
106                             {
107         return average;
108     }
```

5.4.3.5 getMinimum()

```
T com.jack.motorbikestatistics.DataItem< T >.getMinimum ( )  [inline]
```

Getter for minimum of readings.

Returns

T - Minimum value.

References com.jack.motorbikestatistics.DataItem< T >.minimum.

```
114                          {
115         return minimum;
116     }
```

5.4.3.6 getMaximum()

```
T com.jack.motorbikestatistics.DataItem< T >.getMaximum ( )  [inline]
```

Getter for maximum of readings.

Returns

T - Maximum value.

References com.jack.motorbikestatistics.DataItem< T >.maximum.

```
122                         {
123         return maximum;
124     }
```

5.4.3.7 setCurrent()

```
void com.jack.motorbikestatistics.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

T	- New reading.

References com.jack.motorbikestatistics.DataItem< T >.add(), com.jack.motorbikestatistics.DataItem< T >.current, com.jack.motorbikestatistics.DataItem< T >.divide(), com.jack.motorbikestatistics.DataItem< T >.greaterThan(), and com.jack.motorbikestatistics.DataItem< T >.lessThan().

```
135                                     {
136         this.current = value;
137 
138         if ((enableAvgMinMax) && (current instanceof Number)) {
139 
140             /* Sets the average */
141             averageCount++;
142             averageSum = add(averageSum, (Number)value);
143             average = divide(averageSum, averageCount);
144 
145             /* Sets the new minimum and maximums if true */
146             if ((minimum == null) || lessThan((Number)current, (Number)
    minimum)) {
147                 minimum = current;
148             }
149             if ((maximum == null) || greaterThan((Number)current, (Number)
    maximum)) {
150                 maximum = current;
151             }
152         }
153     }
```

5.4.3.8 add()

```
Double com.jack.motorbikestatistics.DataItem< T >.add (
            Number a,
            Number b )  [inline], [private]
```

Function to allow addition of numbers with variable types.

Parameters

a	- First operand.
b	- Second operand.

Returns

Double - Sum.

```
161                                         {
162         return new Double(a.doubleValue() + b.doubleValue());
163     }
```

5.4.3.9 divide()

```
Double com.jack.motorbikestatistics.DataItem< T >.divide (
            Number numerator,
            Number denominator )  [inline], [private]
```

Function to allow division of numbers with variable types.

Parameters

numerator	- Numerator of divisior.
denominator	- Denominator of divisor.

Returns

Double - Result of division.

```
171                                                         {
172         return new Double(numerator.doubleValue() / denominator.doubleValue());
173     }
```

5.4.3.10 greaterThan()

```
boolean com.jack.motorbikestatistics.DataItem< T >.greaterThan (
            Number a,
            Number b )  [inline], [private]
```

Function to chcek whether A is greater than B.

Parameters

a	- First operand.
b	- Second operand.

Returns

boolean - Whether A is greater than B.

```
181                                                 {
182         return a.doubleValue() > b.doubleValue();
183     }
```

5.4.3.11 lessThan()

```
boolean com.jack.motorbikestatistics.DataItem< T >.lessThan (
            Number a,
            Number b )  [inline], [private]
```

Function to chcek whether A is less than B.

Parameters

a	- First operand.
b	- Second operand.

Returns

boolean - Whether A is less than B.

```
191                                                 {
192         return a.doubleValue() < b.doubleValue();
193     }
```

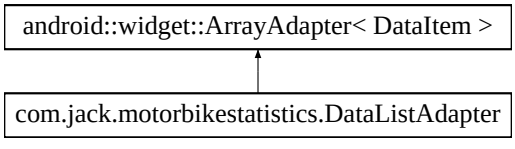
The documentation for this class was generated from the following file:

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

5.5 com.jack.motorbikestatistics.DataListAdapter Class Reference

Adapter class used for displaying statistics.

Inheritance diagram for com.jack.motorbikestatistics.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.

5.5.1 Detailed Description

Adapter class used for displaying statistics.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 DataListAdapter()

```
com.jack.motorbikestatistics.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

cnt	- Context of the adapter to be operating in.
layout↩ ResourceId	- Resource ID for current layout.
data	- ArrayList of statistics to display in ListView.

References com.jack.motorbikestatistics.DataListAdapter.data, and com.jack.motorbikestatistics.DataList↩ Adapter.layoutResourceId.

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

5.5.3 Member Function Documentation

5.5.3.1 getView()

```
View com.jack.motorbikestatistics.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

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

Returns

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

References com.jack.motorbikestatistics.DataItem< T >.getAverage(), com.jack.motorbikestatistics.Data↩ Item< T >.getCurrent(), com.jack.motorbikestatistics.DataItem< T >.getEnabledAvgMinMax(), com.jack.↩ motorbikestatistics.DataItem< T >.getMaximum(), com.jack.motorbikestatistics.DataItem< T >.getMinimum(), and com.jack.motorbikestatistics.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      }
```

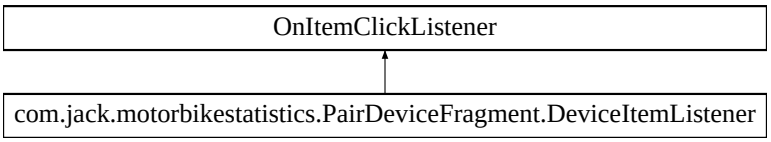
The documentation for this class was generated from the following file:

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

5.6 com.jack.motorbikestatistics.PairDeviceFragment.DeviceItemListener Class Reference

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

Inheritance diagram for com.jack.motorbikestatistics.PairDeviceFragment.DeviceItemListener:



Public Member Functions

- void onItemClick (AdapterView<?> parent, View view, int position, long id)

  Function called when user wants to connect to a device.

5.6.1 Detailed Description

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

5.6.2 Member Function Documentation

5.6.2.1 onItemClick()

```
void com.jack.motorbikestatistics.PairDeviceFragment.DeviceItemListener.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

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

References com.jack.motorbikestatistics.BTDeviceItem.getConnection(), com.jack.motorbikestatistics.BTDeviceItem.getDevice(), com.jack.motorbikestatistics.BTConnection.isConnected(), com.jack.motorbikestatistics.BTDeviceItem.setConnection(), com.jack.motorbikestatistics.BTDeviceItem.setIconID(), and com.jack.motorbikestatistics.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         }
```

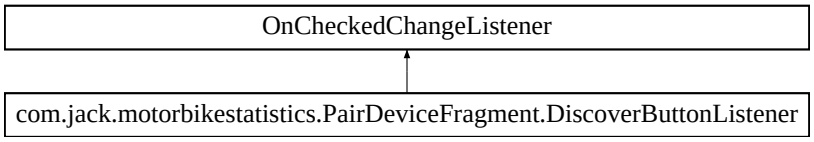
The documentation for this class was generated from the following file:

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

5.7 com.jack.motorbikestatistics.PairDeviceFragment.DiscoverButtonListener Class Reference

Listener for when discovery button is pressed.

Inheritance diagram for com.jack.motorbikestatistics.PairDeviceFragment.DiscoverButtonListener:



Public Member Functions

- void onCheckedChanged (CompoundButton buttonView, boolean isChecked)

  Function for handling when discover toggle button pressed.

5.7.1 Detailed Description

Listener for when discovery button is pressed.

5.7.2 Member Function Documentation

5.7.2.1 onCheckedChanged()

```
void com.jack.motorbikestatistics.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 disover mode.

Parameters

buttonView	- Current view of the toggle button.
isChecked	- 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         }
```

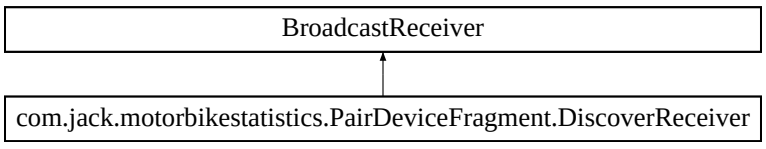
The documentation for this class was generated from the following file:

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

5.8 com.jack.motorbikestatistics.PairDeviceFragment.DiscoverReceiver Class Reference

Receiver for when a new device is discovered.

Inheritance diagram for com.jack.motorbikestatistics.PairDeviceFragment.DiscoverReceiver:



Public Member Functions

- void onReceive (Context context, Intent intent)

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

5.8.1 Detailed Description

Receiver for when a new device is discovered.

5.8.2 Member Function Documentation

5.8.2.1 onReceive()

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

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

Parameters

context	- Context that the application is running in.
intent	- 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",
    BT_DISABLED_ICON);
241                 lvAdapter.add(newDevice);
242                 lvAdapter.notifyDataSetChanged();
243             }
244         }
```

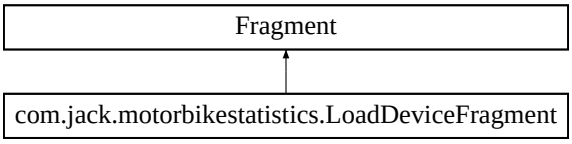
The documentation for this class was generated from the following file:

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

5.9 com.jack.motorbikestatistics.LoadDeviceFragment Class Reference

UI Class for loading saved trips from device.

Inheritance diagram for com.jack.motorbikestatistics.LoadDeviceFragment:



Classes

- class TripItemListener

  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< TripItem > tripList

  List of all trips saved on the logging device.
- ArrayAdapter< TripItem > 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()

```
com.jack.motorbikestatistics.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 com.jack.motorbikestatistics.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 specificed.

Parameters

inflater	- Inflater used to load fragment on UI.
container	- Container where fragment will be shown.
savedInstanceState	- 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 com.jack.motorbikestatistics.LoadDeviceFragment.setBTConnection (
            BTConnection btConnection )  [inline]
```

Setter for current BT connection.

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

Parameters

btConnection	- Logging device bluetooth connection.

References com.jack.motorbikestatistics.LoadDeviceFragment.btConnection.

```
99                                                          {
100         this.btConnection = btConnection;
101     }
```

5.9.3.3 addTrip()

```
final void com.jack.motorbikestatistics.LoadDeviceFragment.addTrip (
            JSONObject jsonData )  [inline], [private]
```

Adds a trip to the ListView specifying name and filesize.

Parameters

jsonData	- 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 com.jack.motorbikestatistics.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.

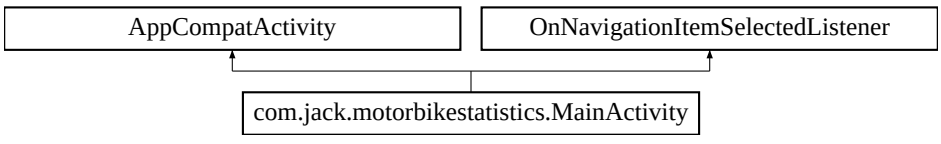
The documentation for this class was generated from the following file:

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

5.10 com.jack.motorbikestatistics.MainActivity Class Reference

Main activity class for fragment navigation.

Inheritance diagram for com.jack.motorbikestatistics.MainActivity:



Public Member Functions

- void onBackPressed ()

  Responsible for closing navigation drawer when back button pressed.
- boolean onNavigationItemSelected (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.

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.
- static RealtimeFragment rtFragment = null

  UI fragment for realtime statistic display.
- static LoadDeviceFragment ldFragment = null

  UI fragment for loading previous trips.
- static PairDeviceFragment pdFragment = null

  UI fragment for pairing to a logging device.

5.10.1 Detailed Description

Main activity class for fragment navigation.

5.10.2 Member Function Documentation

5.10.2.1 onCreate()

```
void com.jack.motorbikestatistics.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

savedInstanceState	- 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      }
```

5.10.2.2 onNavigationItemSelected()

```
boolean com.jack.motorbikestatistics.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

item	- New selected fragment/tab to display.

References com.jack.motorbikestatistics.PairDeviceFragment.getBTConnection(), com.jack.motorbikestatistics.BTConnection.isConnected(), com.jack.motorbikestatistics.MainActivity.ldFragment, com.jack.motorbikestatistics.MainActivity.pdFragment, com.jack.motorbikestatistics.MainActivity.rtFragment, com.jack.motorbikestatistics.LoadDeviceFragment.RXHandler, com.jack.motorbikestatistics.RealtimeFragment.RXHandler, com.jack.motorbikestatistics.LoadDeviceFragment.setBTConnection(), com.jack.motorbikestatistics.BTConnection.setRXHandler(), and com.jack.motorbikestatistics.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     }
```

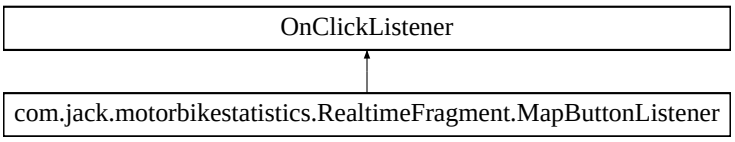
The documentation for this class was generated from the following file:

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

5.11 com.jack.motorbikestatistics.RealtimeFragment.MapButtonListener Class Reference

Listener for starting a map activity when button pressed.

Inheritance diagram for com.jack.motorbikestatistics.RealtimeFragment.MapButtonListener:



Public Member Functions

- void onClick (View v)

  Function for handling when map button pressed.

5.11.1 Detailed Description

Listener for starting a map activity when button pressed.

5.11.2 Member Function Documentation

5.11.2.1 onClick()

```
void com.jack.motorbikestatistics.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.

```
193                                 {
194             Intent intent = new Intent(getActivity(), MapsActivity.class);
195             intent.putExtra("JSONList", jsonList);
196             startActivity(intent);
197         }
```

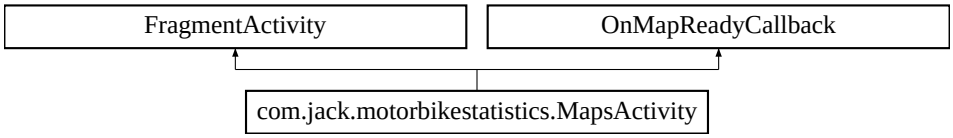
The documentation for this class was generated from the following file:

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

5.12 com.jack.motorbikestatistics.MapsActivity Class Reference

Maps activity class for displaying map data.

Inheritance diagram for com.jack.motorbikestatistics.MapsActivity:



Classes

- class StatisticWindowAdapter

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

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

- boolean getJSONObjects ()

  Gets point data and convert to array of JSON objects.
- 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.

Private Attributes

- GoogleMap mMap

  Google maps object for plotting.
- ArrayList< JSONObject > jsonList = new ArrayList<JSONObject>()

  ArrayList holding all trip data.

5.12.1 Detailed Description

Maps activity class for displaying map data.

5.12.2 Member Function Documentation

5.12.2.1 onCreate()

```
void com.jack.motorbikestatistics.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

savedInstanceState	- Information holding last previous state.

References com.jack.motorbikestatistics.MapsActivity.getJSONObjects().

```
56                                                  {
57         super.onCreate(savedInstanceState);
58         setContentView(R.layout.activity_maps);
59         // Obtain the SupportMapFragment and get notified when the map is ready to be used.
60         SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
61                 .findFragmentById(R.id.map);
62
63         getJSONObjects();
64
65         mapFragment.getMapAsync(this);
66     }
```

5.12.2.2 getJSONObjects()

```
boolean com.jack.motorbikestatistics.MapsActivity.getJSONObjects ( )  [inline], [private]
```

Gets point data and convert to array of JSON objects.

Gets an arraylist of strings passed via a bundle to this activity. These strings are there converted back to JSON objects which will be used for plotting. The reason for not passing straight JSON objects is because they are not serializable and passable between activities.

Returns

boolean - Whether all objects were able to be created.

```
80     {
81         boolean result = true;
82
83         /*
84          * Get our serialized arrayList of jsonStrings
85          * then convert them back to jsonObjects
86          */
87         ArrayList<String> jsonStrings = (ArrayList<String>)getIntent().getSerializableExtra("JSONList");
88         for (int i = 0; i < jsonStrings.size(); i++)
89         {
90             try
91             {
92                 JSONObject jsonObject = new JSONObject(jsonStrings.get(i));
93                 jsonList.add(jsonObject);
94             }
95             catch (JSONException e)
96             {
97                 result = false;
98             }
99         }
100
101         return result;
102     }
```

5.12.2.3 findJSONByLatLng()

```
JSONObject com.jack.motorbikestatistics.MapsActivity.findJSONByLatLng (
             LatLng position )  [inline], [private]
```

Finds JSONObject from ArrayList via LAT/LNG coordinates.

Parameters

position	- Latitude and Longitude position.

Returns

JSONObject - The found JSON object.

References gpsJSON.

```
110                                                     {
111         JSONObject result = null;
112 
113         for (int i = 0; i < jsonList.size(); i++) {
114             JSONObject tmpJSON = jsonList.get(i);
115 
116             try {
117                 JSONObject gpsJSON = tmpJSON.getJSONObject("gps");
118 
119                 Double latitude = gpsJSON.getDouble("lat");
120                 Double longitude = gpsJSON.getDouble("lng");
121 
122                 /* Check to see if latitude and logitudes match */
123                 if ((latitude == position.latitude) && (longitude == position.longitude)) {
124                     result = tmpJSON;
125                     break;
126                 }
127 
128             } catch (JSONException e) {
129                 /* Do nothing */
130             }
131         }
132 
133         return result;
134     }
```

5.12.2.4 calcDistance()

```
float com.jack.motorbikestatistics.MapsActivity.calcDistance (
              LatLng start,
              LatLng end )  [inline], [private]
```

Calculates the absolute distance between two points.

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

Parameters

start	- Start position.
end	- End position.

Returns

flaot - Distance between points in metres.

```
145     {
146         float[] results = new float[1];
```

```
147
148         Location.distanceBetween(start.latitude, start.longitude, end.latitude, end.longitude, results);
149         return results[0];
150     }
```

5.12.2.5 onMapReady()

```
void com.jack.motorbikestatistics.MapsActivity.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 SupportMap↩ Fragment. This method will only be triggered once the user has installed Google Play services and returned to the app.

Parameters

googleMap	- Our map object ready to manipulate.

References com.jack.motorbikestatistics.MapsActivity.calcDistance(), and gpsJSON.

```
165                                                 {
166         mMap = googleMap;
167
168         mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
169
170         /* Set our info window adapter class that is shown when marker clicked */
171         mMap.setInfoWindowAdapter(new StatisticWindowAdapter());
172
173         /* If we have no data don't bother plotting points */
174         if (jsonList.size() != 0)
175         {
176             /* lineOpts will store our route */
177             PolylineOptions lineOpts = new PolylineOptions();
178             lineOpts.color(Color.parseColor( "#CC0000FF"));
179             lineOpts.width(5);
180             lineOpts.visible(true);
181
182             try
183             {
184                 LatLng lastMarker = null;
185
186                 /* Plot every point in the our JSONObject array */
187                 for (int i = 0; i < jsonList.size(); i++)
188                 {
189                     JSONObject rootJSON = jsonList.get(i);
190                     JSONObject gpsJSON = rootJSON.getJSONObject("gps");
191
192                     Double lat = gpsJSON.getDouble("lat");
193                     Double lng = gpsJSON.getDouble("lng");
194                     LatLng location = new LatLng(lat, lng);
195
196                     /* Add this location to our trip line */
197                     lineOpts.add(location);
198
199                     /*
200                      * Check if distance between this point and
201                      * last marker is greater than 5m otherwise don't add marker.
202                      * Adding markers every 5 metres prevents the map being spammed with
203                      * thousands of readings.
204                      */
205                     if ((lastMarker == null) || (calcDistance(location, lastMarker) > 5))
206                     {
```

```
207                         /* Only add a marker if the gps data is valid */
208                         if (gpsJSON.getBoolean("gps_valid") == true) {
209                             MarkerOptions markerOptions = new MarkerOptions();
210                             markerOptions.position(location);
211                             markerOptions.title("Reading: " + i);
212
213                             mMap.addMarker(markerOptions);
214
215                             lastMarker = location;
216
217                             /* Changes camera to point to newest marker */
218                             mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(location, 12));
219                         }
220                     }
221
222                 }
223                 mMap.addPolyline(lineOpts);
224             }
225             catch (JSONException e)
226             {
227                 /* Do nothing */
228             }
229         }
230     }
```

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

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

5.13 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.

5.13.1 Detailed Description

Class for dealing with Orientation functionality on logging device.

5.13.2 Member Function Documentation

5.13.2.1 convertRawAccel()

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

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

Parameters

aRaw	- Raw accelerometer axis value.

Returns

float - Processed acceleration axis in G.

References ACCEL_RANGE.

```
116 {
117   /*
118    * Since using 2G range.
119    * -2G maps to raw value of -32768
120    * +2G maps to raw value of +32767
121    */
122    float a = (aRaw * (float)ACCEL_RANGE) / 32768.0;
123    return a;
124 }
```

5.13.2.2 convertRawGyro()

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

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

Parameters

gRaw	- Raw gyroscope axis value.

Returns

float - Processed rotation axis in deg/sec.

References GYRO_RANGE.

```
133 {
134   /*
135    * since we are using 250 degrees/seconds range
136    * -250 maps to a raw value of -32768
137    * +250 maps to a raw value of 32767
138    */
139   float g = (gRaw * (float)GYRO_RANGE) / 32768.0;
140   return g;
141 }
```

5.13.2.3 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.

```
44 {
45   /* Set up the Gyroscope + Accelerometer */
46   CurieIMU.begin();
47   CurieIMU.setGyroRate(IMU_FREQUENCY);
48   CurieIMU.setAccelerometerRate(IMU_FREQUENCY);
49   CurieIMU.setAccelerometerRange(ACCEL_RANGE);
50   CurieIMU.setGyroRange(GYRO_RANGE);
51
52   IMUfilter.begin(IMU_FREQUENCY);
53 }
```

5.13.2.4 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.

```
69 {
70   static const unsigned long US_PER_READING = 1000000 / IMU_FREQUENCY;
71   static unsigned long usPrevious = micros();
72
73   bool result = false;
74   int accel_raw[NUMBER_AXIS];
75   int gyro_raw[NUMBER_AXIS];
76   float accel_g[NUMBER_AXIS];
77   float gyro_ds[NUMBER_AXIS];
78   unsigned long usNow;
79
80   /* Ensures we stick to the sample rate (by not sampling too early) */
81   usNow = micros();
82   if ((usNow - usPrevious) >= US_PER_READING)
83   {
84     /* Read raw data from the IMU */
85     CurieIMU.readMotionSensor(accel_raw[AXIS_X], accel_raw[AXIS_Y], accel_raw[
     AXIS_Z],
86                               gyro_raw[AXIS_X], gyro_raw[AXIS_Y], gyro_raw[AXIS_Z]);
87
88     /* Convert raw readings from IMU to accel (G) and rotation vel (deg/s) */
89     accel_g[AXIS_X] = convertRawAccel(accel_raw[AXIS_X]);
90     accel_g[AXIS_Y] = convertRawAccel(accel_raw[AXIS_Y]);
91     accel_g[AXIS_Z] = convertRawAccel(accel_raw[AXIS_Z]);
92     gyro_ds[AXIS_X] = convertRawGyro(gyro_raw[AXIS_X]);
93     gyro_ds[AXIS_Y] = convertRawGyro(gyro_raw[AXIS_Y]);
94     gyro_ds[AXIS_Z] = convertRawGyro(gyro_raw[AXIS_Z]);
95
96     /* Update the filter. Orientation is calculated here */
97     IMUfilter.updateIMU(gyro_ds[AXIS_X], gyro_ds[AXIS_Y], gyro_ds[AXIS_Z],
98                         accel_g[AXIS_X], accel_g[AXIS_Y], accel_g[AXIS_Z]);
99
100     /* Increment previous counter */
101     usPrevious += US_PER_READING;
102
103     result = true;
104   }
105
106   return result;
107 }
```

5.13.2.5 getYaw()

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

Returns the Yaw orientation of the device.

Returns

float - Yaw orientation.

References IMUfilter.

```
148 {
149   return IMUfilter.getYaw();
150 }
```

5.13.2.6 getPitch()

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

Returns the Pitch orientation of the device.

Returns

float - Pitch orientation.

References IMUfilter.

```
157 {
158   return IMUfilter.getPitch();
159 }
```

5.13.2.7 getRoll()

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

Returns the Roll orientation of the device.

Returns

float - Roll orientation.

References IMUfilter.

```
166 {
167   return IMUfilter.getRoll();
168 }
```

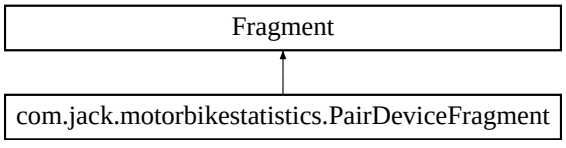
The documentation for this class was generated from the following files:

- logging-device/Orientation.h
- logging-device/Orientation.cpp

5.14 com.jack.motorbikestatistics.PairDeviceFragment Class Reference

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

Inheritance diagram for com.jack.motorbikestatistics.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.

5.14.1 Detailed Description

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

5.14.2 Constructor & Destructor Documentation

5.14.2.1 PairDeviceFragment()

```
com.jack.motorbikestatistics.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     }
```

5.14.3 Member Function Documentation

5.14.3.1 onCreateView()

```
View com.jack.motorbikestatistics.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

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

Returns

View - The UI view of this fragment.

References com.jack.motorbikestatistics.BTDeviceItem.getConnection(), com.jack.motorbikestatistics.PairDeviceFragment.getNeededPrivileges(), com.jack.motorbikestatistics.BTConnection.isConnected(), and com.jack.motorbikestatistics.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,
    btDeviceList);
121         lvDevices.setAdapter(lvAdapter);
122
123         /* Check and set up bluetooth adapter */
124         if (btAdapter == null)
125         {
126             Toast.makeText(getActivity().getApplicationContext(),
127                     "This device has no bluetooth adapter", Toast.LENGTH_LONG).show();
128         }
129         else
130         {
131             /* Check to see if connected device still is connected */
132             if (btConnectedDevice != null)
133             {
134                 if (!btConnectedDevice.getConnection().
    isConnected() ||
135                         !btConnectedDevice.getConnection().
    isRunning())
136                 {
137                     btConnectedDevice = null;
138                 }
139             }
140
141             /* firstRun check to list from being re-populated */
142             if (firstRun)
143             {
144                 firstRun = false;
145
146                 /* Enable bluetooth adapter if disabled */
147                 if (!btAdapter.isEnabled())
148                 {
149                     Intent enableBT = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
150                     startActivityForResult(enableBT, REQUEST_BLUETOOTH);
151                 }
152
153                 while (!btAdapter.isEnabled())
154                 {
155                     /* Wait for BT to be enabled */
156                 }
157
158                 /* Add all paired devices to list */
159                 Set<BluetoothDevice> pairedDevices = btAdapter.getBondedDevices();
160                 if (pairedDevices.size() > 0)
161                 {
162                     for (BluetoothDevice device : pairedDevices)
163                     {
164                         BTDeviceItem newDevice =
165                             new BTDeviceItem(device, "paired", BT_DISABLED_ICON);
166                         btPairedList.add(newDevice);
167                     }
168                 }
169                 btDeviceList.addAll(btPairedList);
170             }
171
172         }
173
174         return myView;
175     }
```

5.14.3.2 getBTConnection()

```
BTConnection com.jack.motorbikestatistics.PairDeviceFragment.getBTConnection ( )  [inline]
```

Getter for getting current connected device.

Returns

BTConnection - Bluetooth device (logging device).

References com.jack.motorbikestatistics.BTDeviceItem.getConnection().

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

5.14.3.3 getNeededPrivileges()

```
void com.jack.motorbikestatistics.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
    .PERMISSION_GRANTED);
212         }
213
214         if (!permsGranted)
215         {
216             ActivityCompat.requestPermissions(getActivity(), permsToRequest, REQUEST_CODE);
217         }
218     }
```

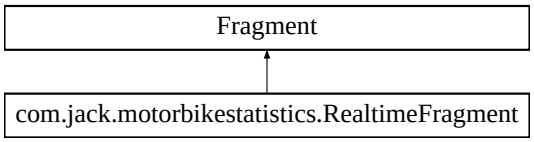
The documentation for this class was generated from the following file:

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

5.15 com.jack.motorbikestatistics.RealtimeFragment Class Reference

UI Class for viewing data sent from the logging device.

Inheritance diagram for com.jack.motorbikestatistics.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.
- ArrayList< String > jsonList

  Array that holds serialised trip data to pass to map.
- 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.

5.15.1 Detailed Description

UI Class for viewing data sent from the logging device.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 RealtimeFragment()

```
com.jack.motorbikestatistics.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.

```
62                              {
63          jsonList = new ArrayList<String>();
64
65          dataList = new SetOfDataItems();
66
67          /* Set up our data items that we will want to log */
68          dataList.add(new DataItem<Double>("Yaw", true));
69          dataList.add(new DataItem<Double>("Pitch", true));
70          dataList.add(new DataItem<Double>("Roll", true));
71          dataList.add(new DataItem<Boolean>("GPS Valid", false));
72          dataList.add(new DataItem<Integer>("Satellites", false));
73          dataList.add(new DataItem<Double>("Latitude", false));
74          dataList.add(new DataItem<Double>("Longitude", false));
75          dataList.add(new DataItem<Double>("Velocity (MPH)", true));
76          dataList.add(new DataItem<Double>("Altitude (FT)", true));
77          dataList.add(new DataItem<Boolean>("Date Valid", false));
78          dataList.add(new DataItem<String>("Date", false));
79          dataList.add(new DataItem<String>("Time", false));
80      }
```

5.15.3 Member Function Documentation

5.15.3.1 onCreateView()

```
View com.jack.motorbikestatistics.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

inflater	- Inflater used for displaying view.
container	- Container that the view will be displayed on.
savedInstanceState	- 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      }
```

5.15.3.2 newData()

```
final void com.jack.motorbikestatistics.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 retreive the data from these child nodes.

Parameters

jsonData	- Received JSONObject from receive handler.

References com.jack.motorbikestatistics.SetOfDataItems.getItemByName(), and com.jack.motorbikestatistics.↩ DataItem< T >.setCurrent().

```
125                                                 {
126
127          try {
128              JSONObject orientObject = jsonData.getJSONObject("orientation");
129              JSONObject gpsObject = jsonData.getJSONObject("gps");
130              JSONObject timeObject = jsonData.getJSONObject("time");
131
132              /* Update our data items */
```

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

5.15.4 Member Data Documentation

5.15.4.1 RXHandler

```
final Handler com.jack.motorbikestatistics.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.

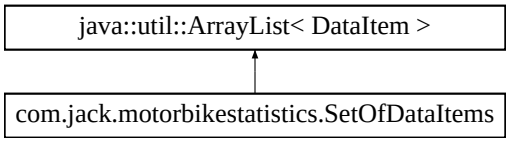
The documentation for this class was generated from the following file:

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

5.16 com.jack.motorbikestatistics.SetOfDataItems Class Reference

ArrayList extension to allow searching via item name.

Inheritance diagram for com.jack.motorbikestatistics.SetOfDataItems:



Public Member Functions

- SetOfDataItems ()

  Constructor, just calls inhereted constructor method.
- DataItem getItemByName (String name)

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

5.16.1 Detailed Description

ArrayList extension to allow searching via item name.

5.16.2 Member Function Documentation

5.16.2.1 getItemByName()

```
DataItem com.jack.motorbikestatistics.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

name	- Name to match.

Returns

DataItem - The item with matching name.

References com.jack.motorbikestatistics.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      }
```

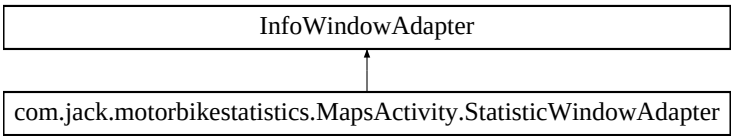
The documentation for this class was generated from the following file:

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

5.17 com.jack.motorbikestatistics.MapsActivity.StatisticWindowAdapter Class Reference

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

Inheritance diagram for com.jack.motorbikestatistics.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.

5.17.1 Detailed Description

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

5.17.2 Member Function Documentation

5.17.2.1 getInfoContents()

```
View com.jack.motorbikestatistics.MapsActivity.StatisticWindowAdapter.getInfoContents (
            Marker marker )  [inline]
```

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

Parameters

marker	- The marker the user has clicked on.

Returns

View - Updated view showing information.

References com.jack.motorbikestatistics.MapsActivity.findJSONByLatLng(), gpsJSON, orientJSON, and timeJSON.

```
254                                                 {
255
256             View v =  getLayoutInflater().inflate(R.layout.map_marker_info, null);
257
258             /* Get latitude and longitude from marker */
259             LatLng latlng = marker.getPosition();
260
261             /* Find the JSONObject relating to this location */
262             JSONObject rootJSON = findJSONByLatLng(latlng);
263             if (rootJSON != null) {
264                 try {
265                     JSONObject gpsJSON = rootJSON.getJSONObject("gps");
266                     JSONObject orientJSON = rootJSON.getJSONObject("orientation");
267                     JSONObject timeJSON = rootJSON.getJSONObject("time");
268
269                     /* Set latitude and longitude in info window */
270                     TextView tvLatLng = (TextView)v.findViewById(R.id.map_latlng);
271                     tvLatLng.setText("Lat/Lng: " + Double.toString(latlng.latitude) + "/"
272                             + Double.toString(latlng.longitude));
273
274                     /* Set time */
275                     TextView tvTime = (TextView)v.findViewById(R.id.map_time);
276                     Calendar cal = Calendar.getInstance();
277                     cal.clear();
278                     cal.set(Calendar.YEAR, timeJSON.getInt("year"));
279                     cal.set(Calendar.MONTH, timeJSON.getInt("month"));
280                     cal.set(Calendar.DATE, timeJSON.getInt("day"));
281
282                     cal.set(Calendar.HOUR, timeJSON.getInt("hour"));
283                     cal.set(Calendar.MINUTE, timeJSON.getInt("minute"));
284                     cal.set(Calendar.SECOND, timeJSON.getInt("second"));
285                     cal.set(Calendar.MILLISECOND, timeJSON.getInt("centiseconds") * 10);
286
287                     /* Create format for date and times then add to view */
288                     DateFormat dateFormat = new SimpleDateFormat("dd/MM/yy HH:mm:ss.SS");
289                     tvTime.setText("Time: " + dateFormat.format(cal.getTime()));
290
291                     /* Velocity & Altitude */
292                     TextView tvVelocity = (TextView)v.findViewById(R.id.map_velocity);
293                     tvVelocity.setText("Velocity: " + gpsJSON.getDouble("vel_mph") + "mph");
294
295                     TextView tvAltitude = (TextView)v.findViewById(R.id.map_altitude);
296                     tvAltitude.setText("Altitude: " + gpsJSON.getDouble("alt_ft") + "ft");
297
298                     /* Orientation */
299                     TextView tvPitch = (TextView)v.findViewById(R.id.map_pitch);
300                     tvPitch.setText("Pitch Angle: " + orientJSON.getDouble("pitch") + "\u00b0");
```

```
301
302                     TextView tvRoll = (TextView)v.findViewById(R.id.map_roll);
303                     tvRoll.setText("Roll/Lean Angle: " + orientJSON.getDouble("roll") + "\u00b0");
304
305                 } catch (JSONException e) {
306                     marker.hideInfoWindow();
307                 }
308             } else {
309                 /* If unable to find relating we hide the info window */
310                 marker.hideInfoWindow();
311             }
312
313             return v;
314         }
```

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

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

5.18 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)

5.18.1 Detailed Description

Class for storing & retrieving data on the logging device.

5.18.2 Member Function Documentation

5.18.2.1 init()

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

Initialisation function for storage module.

Responsible for starting the uSD library.

References USD_CS.

```
38 {
39   SD.begin(USD_CS);
40 }
```

5.18.2.2 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

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

Returns

bool - Whether saving was a success.

References fileName.

```
53 {
54   bool result = false;
55
56   /* Create handle to log file */
57   File logHandle = SD.open(fileName, FILE_WRITE);
58
59   /* If handle exists print line to file */
60   if (logHandle)
61   {
62
63     /* Print line, option to add newline characters */
64     logHandle.print(data);
65     if (newLine)
```

```
66     {
67       logHandle.println();
68     }
69
70     logHandle.close();
71     result = true;
72   }
73   return result;
74 }
```

5.18.2.3 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-existant 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.

```
85 {
86   bool result = false;
87   int i = 0;
88
89   for (i = 0; i < MAX_LOG_FILES; i++)
90   {
91     /* Clear name of log file */
92     memset(fileName, 0, strlen(fileName));
93
94     /* Set the new log file name to: trip_XXXXX.json */
95     sprintf(fileName, "%s%d.%s", LOG_NAME, i, LOG_EXTENSION);
96
97     if (!SD.exists(fileName))
98     {
99       /* If a file doesn't exist */
100       result = true;
101       break;
102     }
103   }
104
105   return result;
106 }
```

5.18.2.4 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.

```
114 {
115   bool filesRemaining = true;
116
117   File root = SD.open("/");
118
119   /* Try to open directory for logs */
120   if (root)
121   {
122     /* Ensure starting from start of directory */
123     root.rewindDirectory();
124
125     while (filesRemaining == true)
126     {
127       /* Try open handle for next file */
128       File entry = root.openNextFile();
129       if (entry)
130       {
131         if (entry.isDirectory() == false)
132         {
133           /* Print out file name & size */
134           fileJSON["name"] = entry.name();
135           fileJSON["size"] = entry.size();
136
137           fileJSON.printTo(BT_SERIAL);
138           BT_SERIAL.println();
139         }
140         entry.close();
141       }
142       else
143       {
144         /* No more files remaining in directory */
145         filesRemaining = false;
146       }
147     }
148
149     root.close();
150   }
151 }
```

5.18.2.5 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.

```
161 {
162   bool nameComplete = false;
163   String fileToOpen = "";
164
165   while (nameComplete == false)
166   {
167     /* Keep reading input in serial until file name is found */
168     if (BT_SERIAL.available() > 0)
169     {
170       char recvByte = BT_SERIAL.read();
171       fileToOpen += recvByte;
172
173       /* Wait until extension is found, then we know full file name */
174       if (fileToOpen.endsWith(LOG_EXTENSION))
175       {
176         nameComplete = true;
177       }
178     }
179   }
180
181   /* Check if file exists */
182   if (SD.exists(fileToOpen))
183   {
184     /* Open file, then read out data byte by byte */
```

```
185     File handle = SD.open(fileToOpen);
186     if (handle)
187     {
188
189       while (handle.available())
190       {
191         char readByte = handle.read();
192
193         BT_SERIAL.write(readByte);
194       }
195
196       handle.close();
197     }
198   }
199 }
```

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

- logging-device/Storage.h
- logging-device/Storage.cpp

5.19 com.jack.motorbikestatistics.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.

5.19.1 Detailed Description

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

5.19.2 Constructor & Destructor Documentation

5.19.2.1 TripItem()

```
com.jack.motorbikestatistics.TripItem.TripItem (
            String name,
            int size )  [inline]
```

Constructor for creating of a TripItem.

Sets the original file name and size.

Parameters

name	- Trip name.
size	- Size of the file.

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

5.19.3 Member Function Documentation

5.19.3.1 getTripName()

```
String com.jack.motorbikestatistics.TripItem.getTripName ( )  [inline]
```

Getter for trip name.

Returns

String - Trip name.

References com.jack.motorbikestatistics.TripItem.tripName.

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

5.19.3.2 setTripName()

```
void com.jack.motorbikestatistics.TripItem.setTripName (
            String tripName )  [inline]
```

Setter for trip name.

Parameters

tripName	- New trip name.

References com.jack.motorbikestatistics.TripItem.tripName.

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

5.19.3.3 getFileSize()

```
int com.jack.motorbikestatistics.TripItem.getFileSize ( )  [inline]
```

Getter for trip filesize.

Returns

int - Filesize in bytes.

References com.jack.motorbikestatistics.TripItem.fileSize.

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

5.19.3.4 setFileSize()

```
void com.jack.motorbikestatistics.TripItem.setFileSize (
              int fileSize )  [inline]
```

Setter for trip filesize.

Parameters

fileSize	- New trip filesize.

References com.jack.motorbikestatistics.TripItem.fileSize.

```
64                                      {
65          this.fileSize = fileSize;
66      }
```

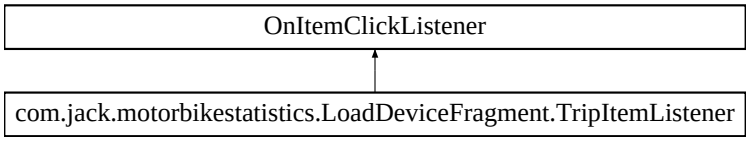
The documentation for this class was generated from the following file:

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

5.20 com.jack.motorbikestatistics.LoadDeviceFragment.TripItemListener Class Reference

Listener used to identify when a trip has been pressed.

Inheritance diagram for com.jack.motorbikestatistics.LoadDeviceFragment.TripItemListener:



Public Member Functions

- void onItemClick (AdapterView<?> parent, View view, int position, long id)

  Loads a trip the user has specified.

5.20.1 Detailed Description

Listener used to identify when a trip has been pressed.

5.20.2 Member Function Documentation

5.20.2.1 onItemClick()

```
void com.jack.motorbikestatistics.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 com.jack.motorbikestatistics.TripItem.getTripName(), com.jack.motorbikestatistics.BTConnection.↩ isConnected(), com.jack.motorbikestatistics.RealtimeFragment.RXHandler, com.jack.motorbikestatistics.BT↩ Connection.setRXHandler(), and com.jack.motorbikestatistics.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         }
```

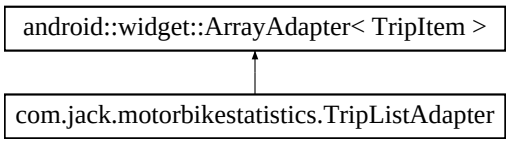
The documentation for this class was generated from the following file:

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

5.21 com.jack.motorbikestatistics.TripListAdapter Class Reference

Adapter class used for displaying all trips.

Inheritance diagram for com.jack.motorbikestatistics.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.

5.21.1 Detailed Description

Adapter class used for displaying all trips.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 TripListAdapter()

```
com.jack.motorbikestatistics.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

cnt	- Context of the adapter to be operating in.
layout↩ ResourceId	- Resource ID for current layout.
data	- ArrayList of statistics to display in ListView.

References com.jack.motorbikestatistics.TripListAdapter.data, and com.jack.motorbikestatistics.TripListAdapter.↩ layoutResourceId.

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

5.21.3 Member Function Documentation

5.21.3.1 getView()

```
View com.jack.motorbikestatistics.TripListAdapter.getView (
            int position,
            View convertView,
            ViewGroup parent )  [inline]
```

Function for returning the view of each list item (TripItem).

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

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

Returns

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

References com.jack.motorbikestatistics.TripItem.getFileSize(), and com.jack.motorbikestatistics.TripItem.getTrip↩ Name().

```
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

5.22 com.jack.motorbikestatistics.DataListAdapter.ViewHolder Class Reference

Class that holds all data displayed for each ListItem.

5.22.1 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

5.23 com.jack.motorbikestatistics.TripListAdapter.ViewHolder Class Reference

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

5.23.1 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

5.24 com.jack.motorbikestatistics.BTDeviceListAdapter.ViewHolder Class Reference

Class that holds all data displayed for each ListItem.

5.24.1 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

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 com.jack.motorbikestatistics.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 seperate 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 com.jack.motorbikestatistics.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 com.jack.motorbikestatistics.BTDeviceListAdapter

  Adapter class used for displaying bluetooth devices.
- class com.jack.motorbikestatistics.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 com.jack.motorbikestatistics.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 com.jack.motorbikestatistics.DataListAdapter

  Adapter class used for displaying statistics.
- class com.jack.motorbikestatistics.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/LoadDeviceFragment.java File Reference

Fragment/Tab for providing UI for loading from device.

Classes

- class com.jack.motorbikestatistics.LoadDeviceFragment
  UI Class for loading saved trips from device.
- class com.jack.motorbikestatistics.LoadDeviceFragment.TripItemListener
  Listener used to identify when a trip has been pressed.

6.6.1 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

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

Main activity class responsible for tabbing.

Classes

- class com.jack.motorbikestatistics.MainActivity
  Main activity class for fragment navigation.

6.7.1 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

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

Maps activity class reponsible for showing data on Google Maps.

Classes

- class com.jack.motorbikestatistics.MapsActivity

  Maps activity class for displaying map data.
- class com.jack.motorbikestatistics.MapsActivity.StatisticWindowAdapter

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

6.8.1 Detailed Description

Maps activity class reponsible 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

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

Fragment/Tab for connecting to the logging device.

Classes

- class com.jack.motorbikestatistics.PairDeviceFragment

  UI Class for discovering, pairing and connecting to the logging device.
- class com.jack.motorbikestatistics.PairDeviceFragment.DiscoverReceiver

  Receiver for when a new device is discovered.
- class com.jack.motorbikestatistics.PairDeviceFragment.DiscoverButtonListener

  Listener for when discovery button is pressed.
- class com.jack.motorbikestatistics.PairDeviceFragment.DeviceItemListener

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

6.9.1 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

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

Fragment/Tab for viewing streamed statistics.

Classes

- class com.jack.motorbikestatistics.RealtimeFragment
  UI Class for viewing data sent from the logging device.
- class com.jack.motorbikestatistics.RealtimeFragment.MapButtonListener
  Listener for starting a map activity when button pressed.

6.10.1 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

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

Extension of ArrayList allows for searching via name.

Classes

- class com.jack.motorbikestatistics.SetOfDataItems

  ArrayList extension to allow searching via item name.

6.11.1 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

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

Class for holding information relating to a specific trip.

Classes

- class com.jack.motorbikestatistics.TripItem

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

6.12.1 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

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

UI ListView adapter to display all saved trips.

Classes

- class com.jack.motorbikestatistics.TripListAdapter

  Adapter class used for displaying all trips.
- class com.jack.motorbikestatistics.TripListAdapter.ViewHolder

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

6.13.1 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

6.14 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 relatime 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.

6.14.1 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

6.14.2 Function Documentation

6.14.2.1 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, Storage::init(), Orientation::init(), LED_PIN, and serGPS().

```
98 {
99   pinMode(LED_PIN, OUTPUT);
100
101   /* Initialise our created modules */
102   storage.init();
103   orientation.init();
104
105   /* Set up serial for wireless data transmission */
106   BT_SERIAL.begin(BT_BAUD);
107
108   /* Set up serial for GPS module */
109   serGPS.begin(GPS_BAUD);
110 }
```

6.14.2.2 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.

```
121 {
122
123   /* Check if mode change character received from front-end */
124   if (BT_SERIAL.available() > 0)
125   {
126     char modeChar = BT_SERIAL.read();
127
128     OPERATING_MODE newMode;
129
130     /* If valid new mode character found change system state */
131     if (parseNewMode(modeChar, newMode) == true)
132     {
133       systemMode = newMode;
134     }
135   }
136
137   /* State machine for choosing what option takes place */
138   switch (systemMode)
139   {
140     case IDLE:
141     {
142       /*
143        * In IDLE mode MCU does nothing.
144        * System waits and still parses incoming commands.
145        */
146       break;
147     }
148
149     case REALTIME:
150     {
151       realTimeMode();
152       break;
153     }
154   }
155 }
```

6.14.2.3 parseNewMode()

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

Returns whether system should change operating mode.

Parameters

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

Returns

bool - Whether a valid command was found.

References IDLE_CHAR.

```
165 {
166   bool result = true;
167
168   switch (modeChar)
169   {
170     case IDLE_CHAR:
171     {
172       newMode = IDLE;
173       break;
174     }
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
185       newMode = REALTIME;
186       break;
187     }
188
189     case LIST_SAVED_CHAR:
190     {
191       /*
192        * Load all trips and send to application.
193        * Once we have finished sending trips we can go back to idle mode.
194        */
195       storage.loadTripNames();
196       newMode = IDLE;
197       break;
198     }
199
200     case LOAD_TRIP_CHAR:
201     {
202       /* Load a specific trip by file name */
203       storage.loadSavedTrip();
204       newMode = IDLE;
205       break;
206     }
207
208     default:
209     {
210       /*
211        * If not a valid operating mode character
212        * then return that parsing failed.
213        */
214       result = false;
215     }
216   }
217
218   return result;
219 }
```

6.14.2.4 realTimeMode()

```
void realTimeMode ( )
```

Responsible for completing work needed in relatime 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, Orientation::pollIMU(), Storage::saveToFile(), and serGPS().

```
234 {
235   static const unsigned int MAX_STRING_SIZE = 512;
236   static const unsigned long PRINT_DELAY = 1000;
237   static unsigned long lastMillis = 0;
238   char jsonString[MAX_STRING_SIZE];
239 
240   /* Poll our IMU to update XYZ */
241   orientation.pollIMU();
242 
243   /* Parse NMEA codes into GPS object */
244   while (serGPS.available() > 0)
245   {
246     gps.encode(serGPS.read());
247   }
248 
249   /* Print orientation and location information */
250   if ((millis() - lastMillis) > PRINT_DELAY)
251   {
252     digitalWrite(LED_PIN, HIGH);
253 
254     addOrientationToJSON();
255     addGPSToJSON();
256     addTimeToJSON();
257 
258     /* Print our json object into a string */
259     mainJSON.printTo(jsonString, MAX_STRING_SIZE);
260 
261     /* Log JSON to the microSD */
262     storage.saveToFile(jsonString, true);
263 
264     /* Print to our bluetooth module */
265     BT_SERIAL.println(jsonString);
266 
267     lastMillis = millis();
268     digitalWrite(LED_PIN, LOW);
269   }
270 }
```

6.14.2.5 addOrientationToJSON()

```
void addOrientationToJSON ( )
```

Responsible for updating orientation JSON object with newest information.

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

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

```
280 {
281   orientJSON["yaw"] = orientation.getYaw();
282   orientJSON["pitch"] = orientation.getPitch();
283   orientJSON["roll"] = orientation.getRoll();
284 }
```

6.14.2.6 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.

```
295 {
296   /* Add location information */
297   gpsJSON["gps_valid"] = gps.location.isUpdated();
298   gpsJSON["lat"] = double_with_n_digits(gps.location.lat(), 6);
299   gpsJSON["lng"] = double_with_n_digits(gps.location.lng(), 6);
300
301   /* Other crucial GPS information */
302   gpsJSON["available"] = gps.satellites.value();
303   gpsJSON["vel_mph"] = gps.speed.mph();
304   gpsJSON["alt_ft"] = gps.altitude.feet();
305 }
```

6.14.2.7 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.

```
317 {
318   /* Add time information to JSON */
319   timeJSON["time_valid"] = gps.date.isValid() && gps.time.isValid();
320   timeJSON["day"] = gps.date.day();
321   timeJSON["month"] = gps.date.month();
322   timeJSON["year"] = gps.date.year();
323
324   timeJSON["hour"] = gps.time.hour();
325   timeJSON["minute"] = gps.time.minute();
326   timeJSON["second"] = gps.time.second();
327   timeJSON["centiseconds"] = gps.time.centisecond();
328 }
```

6.15 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 acelerometer +-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.

6.15.1 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).

6.16 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)

6.16.1 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.