We have two options to get the EEG data to our android device. We could either call the BufferClient (which runs on a pc) from our android device, or we could run the BufferClient on our android. Both options have problems.
If we run BufferClient from our android we need to determine how to get BufferClient its data. We have essential just shifted the problem one jar earlier. If we run BufferClient on our computer we have to find a way to call a .jar on another system (i.e. call from android to pc) while maintaining control.

The issues with either solution are exacerbated by the fact that android can't normally call jar files. It seems (at least to me, Peter) that the most viable option is to make our own version of BufferClient inside our sigViewer, since we have the the .java files we can co-opt most of their code (or even all of it, if we're lucky). An alternative is to run the jar from android anyway using a workaround, but this might seriously hamper our ability to get it to communicate with our android application, let alone a server on the computer.

As for how the data is currently flowing, the BufferClient opens a socket (listens to a specific ip address and port) and pulls in binary data. This data is then sent to sigviewer.m through buffer.m. The transfer happens over TCP (as opposed to UDP). The reason for TCP is probably because TCP works in discrete packets, which makes staying synchronized easier. They do not use TCP as say a website would, and have disabled the delay (Nagle's algorithm) that's normally enabled. This function waits with sending a new packet until it's filled, making for more efficient but slower data transfer. They also drastically lower the timeout (500ms, instead of 1500-6000ms). This is not a problem, so long as we make our version do the same. The reason behind these changes have to do with throughput, the faster the (comparatively tiny) packets are received the better the resolution in the time domain.

The input output document states that getDoubleData is used but that we could use float instead. This should not be necessary, the difference in get<Datatype>Data are almost exclusively for deciding what the the datatype casting, and we could always simply re-cast this if necessary. Since a double is effectively a double precision (i.e. 64 instead of 32 bits) float, there is no issue.

It is worth looking into SignalViewer.java. This is a very basic implementation of a SignalViewer; the simplicity and comparatively high code quality make it comparatively easy to parse. It also allows us to see what data we need to create or take in from the buffer. SignalViewer.java also calls on a few other java classes that we might want to look into; BufferClientClock.java (requires some work), Header.java (short and easy), SamplesEventsCount (very short and easy).