## **Processing Simulation**

The design of the Processing simulation is comprised of byte to float conversion, simulation object creation, object translation and rotation and feature creation.

### **Byte to Float Conversion**

Once the orientation data has been transmitted wirelessly, the orientation data that was split into four bytes, needs to be converted back to floating point representation to be of any use to the simulation. The 12 bytes of data (excluding the identification byte) are converted back to the x, y and z orientation angle floating point values. An IMU class was created in the Processing program to store the orientation data to be used to rotate the arm in the simulation. Three instances of the class were created, one for each section of the arm. The orientation data received is stored within one of these instances depending on the identification byte received with the data. The figure below depicts the byte to float conversion process.



Figure : C implementation of the byte to float conversion in Processing.

### **Object Creation**

In order to create the arm in simulation that can be rotated according to the motion of the human arm, 3D objects have to be created and aligned in the simulation. Processing allows 3D objects of .obj format to be imported and used in the program. Therefore, upper arm, forearm and hand objects were created in ‘Autodesk Fusion 360’ that were then exported as .obj files into Processing. When designing the objects, approximate dimensions of the arm were recorded such that the objects that were created are relatively proportional relative to one another. When exporting the created objects, they had to be aligned with the centre of the workspace as this point is treated as the point of rotation around which the object would be rotated in the Processing simulation.

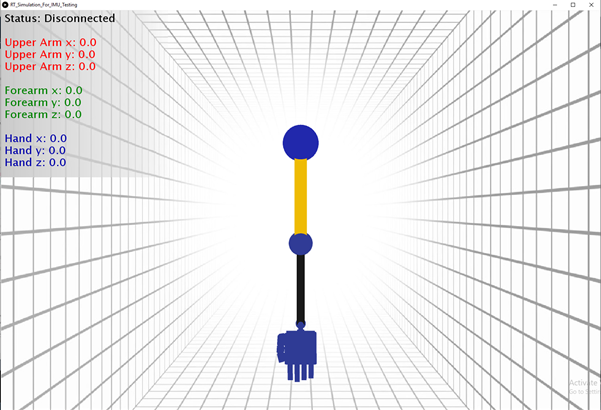
After importing the objects to the Processing program, some needed to be scaled up as they were too small. After scaling the arm objects needed to be aligned to construct the arm the arm that was going to be rotated in the simulation. Other features such as simulation background, display of orientation angles and connection status display was also added to the simulation (see Appendix A for code and images).

Figure : Image depicting the completed simulation.

### **Arm Rotation**

The simulation works by constantly displaying the same objects, background and other features in a loop. When an object is rotated it is simply updated and displayed with the new rotation. When the human arm rotates, the rotations performed by the upper arm, will influence the IMU readings on the forearm and the hand. Because the simulation rotates all three objects in one reference frame and the rotation of the upper arm will rotate the forearm and hand objects, the angle that the upper arm adds to the other two IMU readings when it rotates needs to be subtracted from the readings in order for the consecutive parts of the arm are rotated by the angle which the human hand and forearm have rotated by. If this is not done the forearm in the simulation would be rotated by the angle which the upper arm IMU senses, plus the offset that the upper arm adds to the forearm IMU, plus that actual rotation that the forearm IMU senses when the forearm rotates. This problem gets more severe when it comes to the hand rotations as the offsets of the upper arm and the forearm would be added to the rotation angle of the hand IMU. Figure 3 depicts how this subtraction is implemented in software.

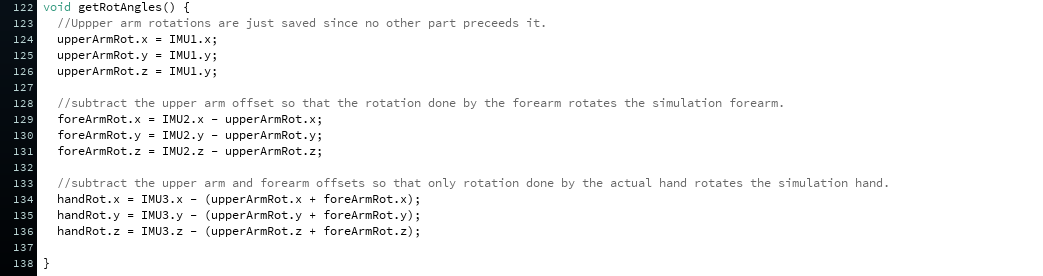


Figure : Subtraction of IMU rotation offsets.

Once these offsets are subtracted from the forearm and the hand IMU orientation angles, the arm in the simulation is rotated by the adjusted angles. Figure 4 depicts the code used to rotate the arm in the simulation.

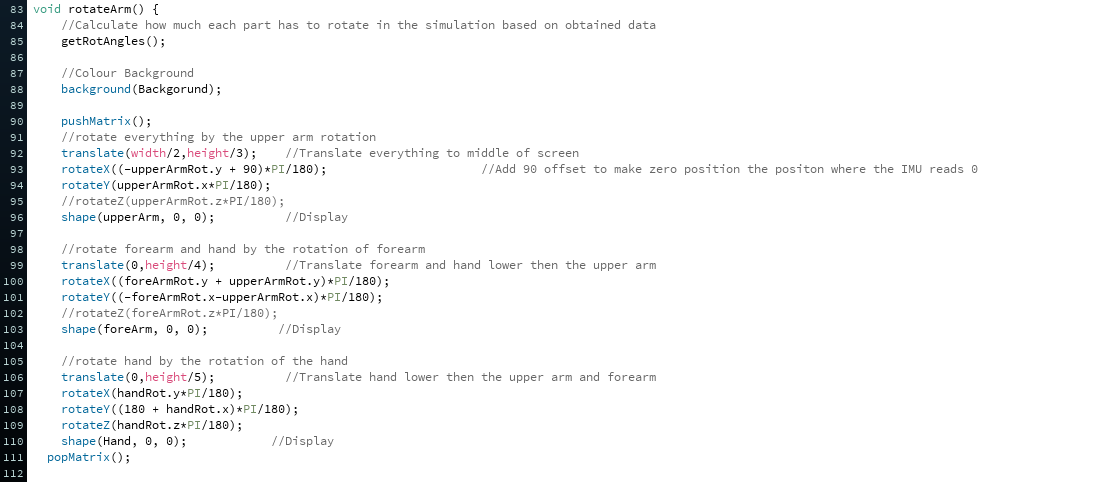


Figure : Processing code used to rotate the simulation components.

## **Appendix A**

Objects and features that are part of the simulation and code used to create the simulation.

