**Part 1: IMU to Unity <2 videos>**

Let’s say our submarine is automatic and has move into an area where we cannot see anything underwater, it is pitch black even with the lights or we would like to visualize the surroundings in 3 dimensions, well our program on unity is able to tell what orientation the submarine is in based on the imu located on board alone.

<Show the video of how we can turn the IMU in any direction and it is reflected in the unity project; can show how even the most extreme changes is reflected, showing the accuracy of the IMU and the code>  
  
These are the assets and code that we have used to do our 1st way of communication between the digital twins

<Introduce each asset in the 1st unity program, from the fbx submarine to the code(which has to be expanded and explained), and other aesthetics like the water>

* (Coming to the ros code, you open it, then explain then show how you start the simulation in ros, with what it does) E.g. here is the ros code, as you can see we are using ubuntu software to uplink ros, the robot and unity. This is where ros receives the input(gyroscopic angular acceleration) from the imu and sends the information to unity to be reflected via the quarternion code.

**Part 2: IMU to Unity <2 videos>**

Let’s say now, our automatic submarine is going to places we do not want, as such we can override their controls with the unity project itself.

<Show the video of how we can turn control the submarine’s thrusters with unity project; in all 6 directions with P as a stop failsafe; show how 1 keypress it will keep going in real life and reflected in the submarine going in a certain direction>  
  
These are the assets and code that we have used to do our 2nd way of communication between the digital twins between the unity project and the thrusters

<Introduce each asset in the 2nd unity program, from the fbx submarine to the code(which has to be expanded and explained), and (to cut time) say the water is imported from the previous project>

* (Coming to the ros code, you open it, then explain then show how you start the simulation in ros, with what it does) E.g. here is the ros code, as you can see we are using ubuntu software to uplink ros, the robot and unity. This is where ros receives the input from the keypresses and sends it to the thrusters of the submarine to be reflected as such.
* When you come to the relay code with all the if statements, explain that because the thrusters are already being set at a fixed speed, we encountered a problem that unity was constantly sending the signal even when the keypress was let go, causing the thrusters to constantly reset as the code signal from unity was coming in at a high frequency. By implementing GetKeypressDown in our code and rigid bodies in Unity, we managed to force the code to send a signal only when it receives a keypress from the user.