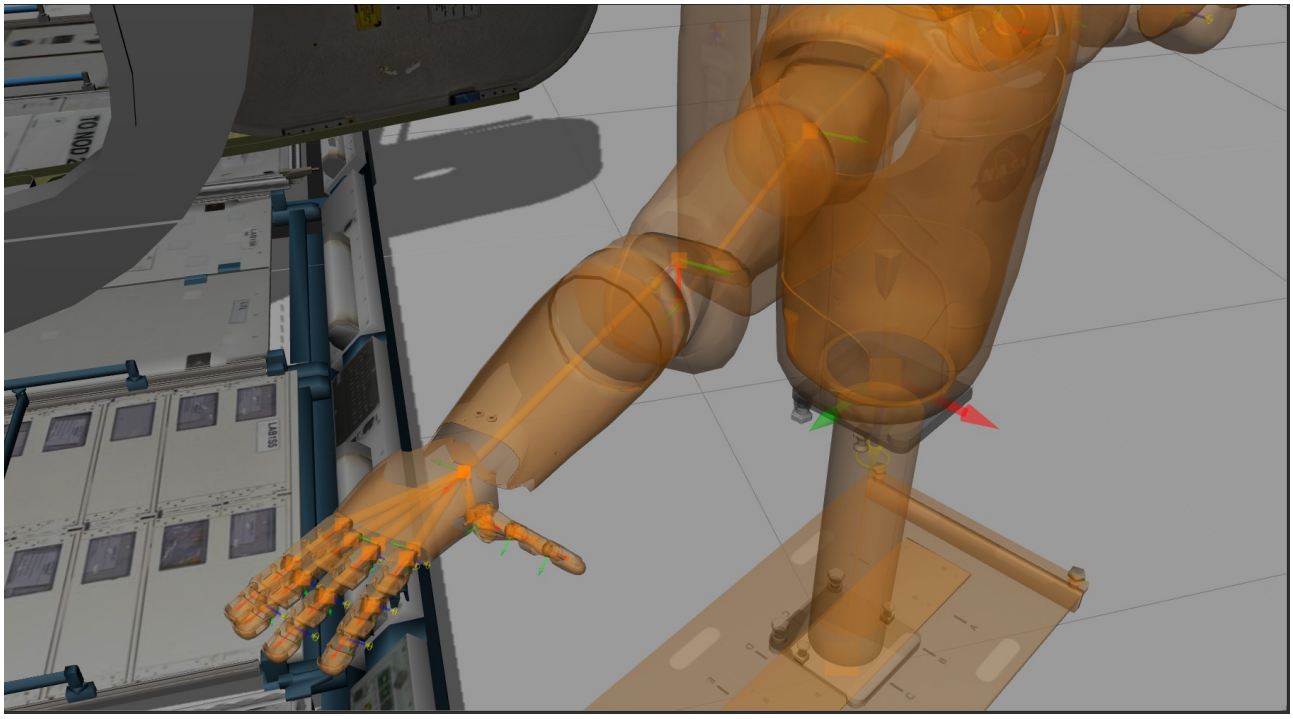
Myo ROS Controller

Χατζηπαράσχης Δημήτρης
2011030039

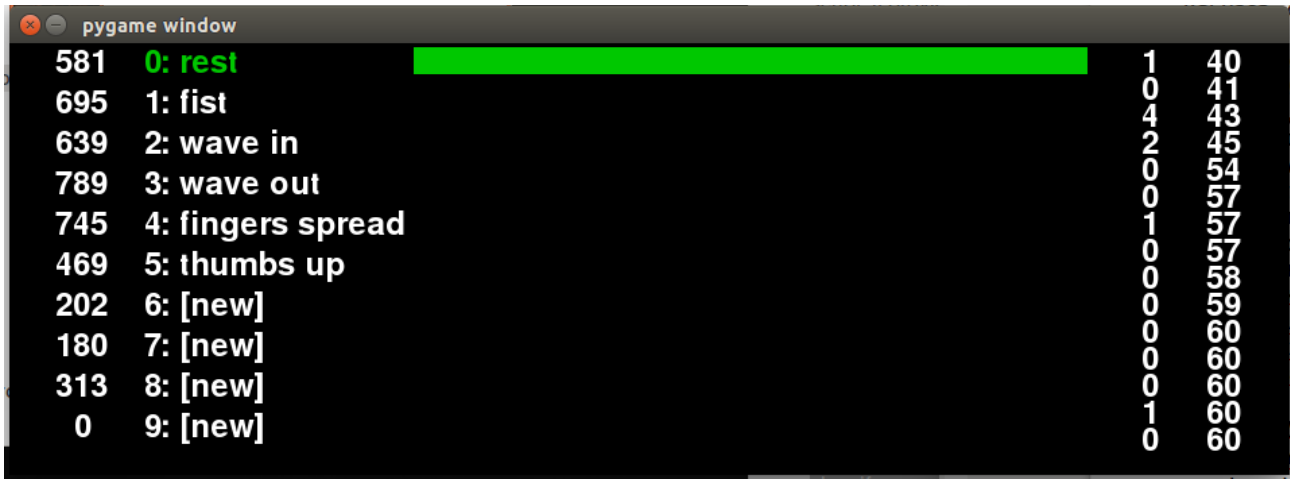
Στόχος του project αυτού ήταν η δημιουργία μιας διεπαφής επικοινωνίας ανάμεσα στο περιβραχιόνιο Myo και το περιβάλλον του GazeboSim, έτσι ώστε να μπορεί να χρησιμοποιηθεί ως ασύρματο χειριστήριο για ένα προσομοιωμένο ρομπότ, μέσα στον κόσμο του Gazebo. Για την υλοποίηση αυτής της επικοινωνίας άλλα και την επεξεργασία των raw δεδομένων που παρείχε το περιβραχιόνιο Myo, χρησιμοποιήθηκε το λογισμικό ROS.

Το ρομποτικό σύστημα που επέλεξα για τη προσομοίωση είναι το Robonaut 2, καθώς ως ανθρωποειδές ρομπότ διαθέτει άκρα (χέρια) τα οποία πλησιάζουν αρκετά στην “προσομοίωση” της κίνησης των ανθρωπίνων χεριών, καθώς διαθέτουν παρόμοιους “συνδετικούς κόμβους” και μέρη. Ο αριθμός, μάλιστα, των συνολικών joints που διαθέτει στο χέρι του είναι 18, πλησιάζοντας τους βαθμούς ελευθερίας του ανθρωπίνου χεριού (όσο αυτό μπορεί να μετρηθεί) και επίσης διαθέτει 7 joints-βαθμους ελευθερίας σε κάθε βραχίωνα του. Επομένως, είναι εμφανές ότι θα μπορεί να προσομοιώσει “οποιαδήποτε” κίνηση-κατάσταση του ανθρωπίνου χεριού του δοθεί ως εντολή. Το Myo από την άλλη μεριά, έχει τη δυνατότητα να μπορεί να μεταδώσει όλη την απαραίτητη πληροφορία για να εξάγουμε σε ποια κατάσταση βρίσκεται το χέρι στο οποίο είναι φορεμένο, από τη χειρονομία του χεριού μέχρι τον προσανατολισμό του βραχίονα και τη μετατόπιση του.



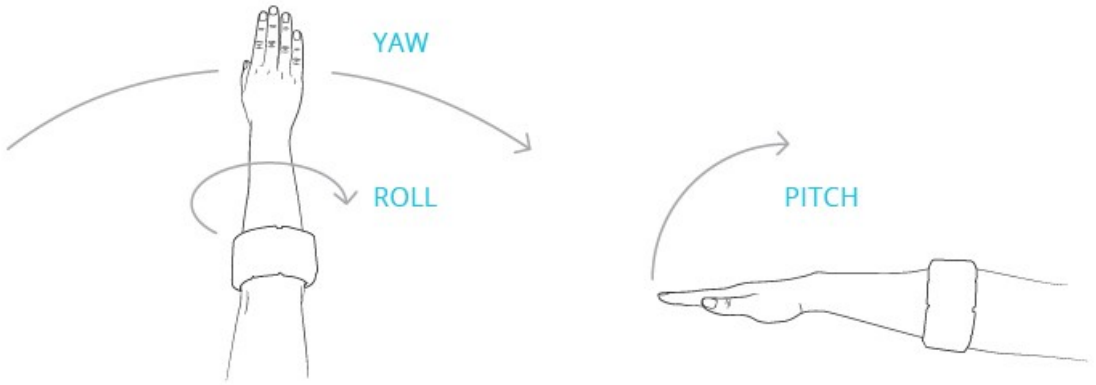
Το μοντέλο κίνησης

Αρχικά, για την απόφαση της σωστής χειρονομίας , την οποία ο χρήστης εκτελεί, έχω συμπεριλάβει μια εφαρμογή από το SDK του Myo η οποία εφαρμόζει training πάνω σε EMG δεδομένα εισόδου που λαμβάνει. Ουσιαστικά η εφαρμογή αυτη μπορεί να εκπαιδευτεί ώστε να μάθει κάποια συγκεκριμένη χειρονομία και να μπορεί να την “ξεχωρίσει” από οποιαδήποτε άλλη.



Από όσο φαίνεται και στην εικόνα, υπάρχουν 9 slots στα οποία μπορεί να αποθηκευθεί οποιαδήποτε χειρονομία που θα θεωρήσουμε εμείς ότι χρειάζεται, δίνοντας ένα μεγάλο εύρος επιλόγων ανάλογα με την εφαρμογή του ρομποτικοί συστήματος που θέλουμε να υλοποιήσουμε. Συγκεκριμένα, έχω συμπεριλάβει τις βασικές τέσσερις χειρονομίες που θεώρησα ότι μπορεί το Robonaut να εκτελεί -συν της rest gesture- , κατά τις οποίες σε συνδυασμό με την παραμετροποίηση των δυνάμεων με τις οποίες θα τις ασκεί (δηλαδή το stiffness των δαχτύλων του), κατάφερα να μπορώ να προσομοιώσω μια μεγάλη γκάμα επιλόγων, για το αν θέλουμε το Robonaut να ενεργήσει άμεσα πραγματοποιώντας μια χειρονομία που θα του δώσουμε ως είσοδο ή θα πρέπει να είναι λιγότερο ευαίσθητο εάν π.χ θέλουμε να πιάσουμε κάτι ή να αποφύγουμε κάτι, μέσα στον κόσμο του Gazebo.

Από την άλλη μεριά, για τη προσομοίωση του κίνησης και του προσανατολισμού του ανθρωπίνου βραχίονα , χρησιμοποίησα τα δεδομένα των IMU αισθητήρων του Myo. Συγκεκριμένα, από τα δεδομένα αυτά εξήγαγα τους αριθμούς Quaternion που έδειχναν το orientation του Myo και τους μετέτρεψα στις 3 γωνίες Euler, Roll, Pitch και Yaw.



Έτσι με αυτόν τον τρόπο, η κίνηση του βραχίονα άλλα και η κατάσταση-χειρονομία του χεριού μπορεί να προσομοιωθεί με μεγάλη επιτυχία και άρα μπορεί να μεταφερθεί ως είσοδος στο ρομποτικό σύστημα για να τη μιμηθεί. Για να πραγματοποιηθεί αυτό, θα πρέπει να δημιουργηθεί μια διεπαφή ανάμεσα στο Myo και το Περιβάλλον του Gazebo, με το προσομοιωμένο Robonaut, ώστε ανάλογα με τα δεδομένα των αισθητήρων του Myo, θα εκτελούνται και οι ανάλογες κλήσεις στα joints του Robonaut για να μπορεί να αλληλεπιδράσει ομοίως με την κίνηση.

Robonaut_myo ROS Package

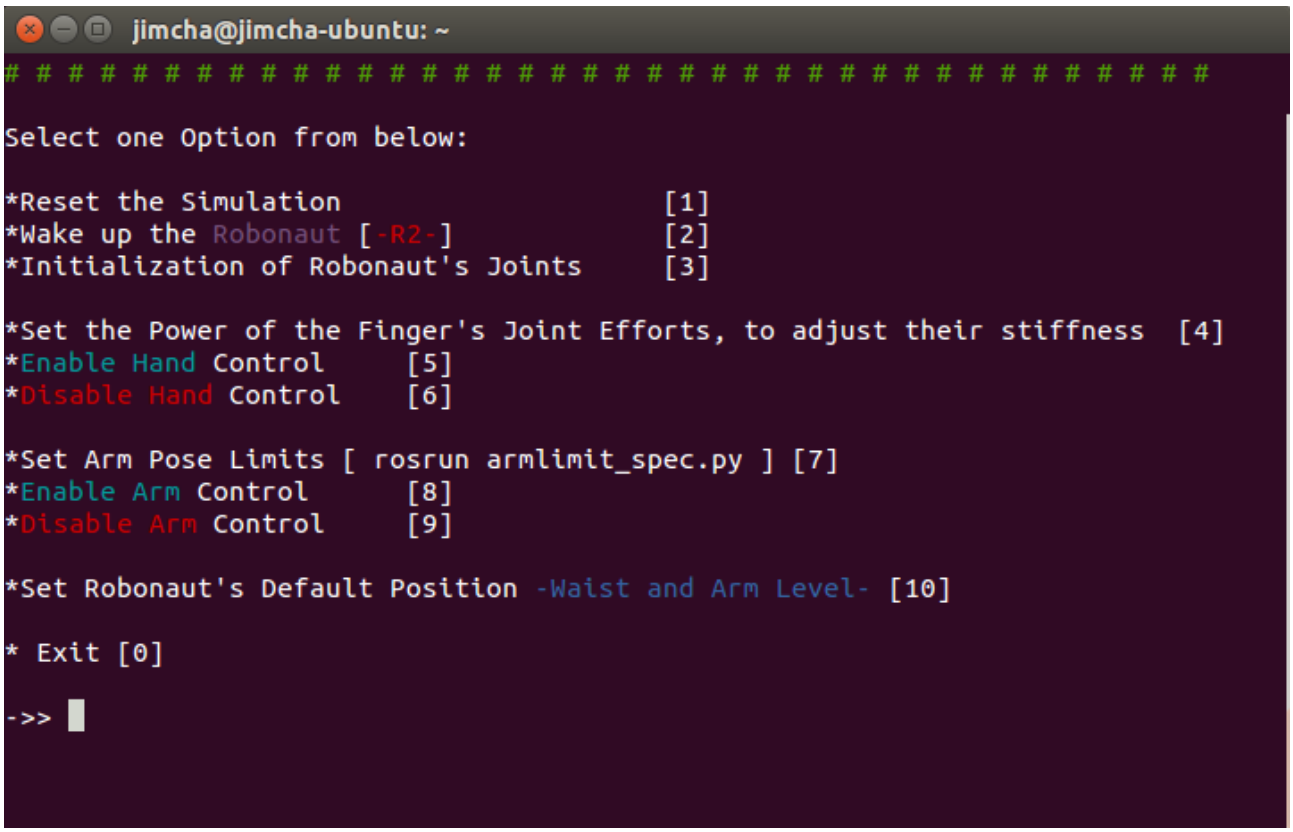
Όπως προανέφερα, για την υλοποίηση της παραπάνω διεπαφής χρησιμοποίησα το λογισμικό του ROS και συγκεκριμένα την εκδοσή Jade (σε Ubuntu 14.04 OS). Με τη βοήθεια του ROS, δημιούργησα από την αρχή ένα package για την διεπαφή αυτήν Myo-Gazebo-Robonaut2 , εν ονόματι **robonaut_myo**, το οποίο πακέτο περιέχει όλους τους κόμβους στους οποίους γίνεται η επεξεργασία των raw data που παρέχει το Myo και η τελική επικοινωνία με τον κόσμο του GazeboSim και συγκεκριμένα το Simulated Robonaut. Τα ROS nodes μεταξύ τους επικοινωνούν μεσώ των ROS Topics, καθώς μπορούν να κάνουν Publish μηνύματα πάνω σε αυτά είτε Subscribe για να διαβάσουν δεδομένα ή μηνύματα από άλλα ROS nodes.

Στο πακέτο αυτό περιλαμβάνονται τα εξής ROS Nodes,

- **Myo_node :** Το οποίο αποτελείται από 3 αρχεία , το myo_raw.py , το myo.py και το βασικότερο όλων classify_myoNode.py. Το τελευταίο

αρχείο υλοποιεί το gesture classification όπως το είχα αναφέρει πρωτύτερα και δημιουργεί 4 ROS Topics , στα οποία εξάγει τα raw data του Myo -myo_imu,myo_emg,myo_arm και myo_handgest. Ο βασικός σκοπός αυτού του node είναι να εντοπίσει και να συνδεθεί με το Myo που βρίσκεται στον περιβάλλοντα χώρο , βάση του πρωτοκόλλου επικοινωνίας του Myo και του Firmware του , επιτρέποντας το να μεταδώσει τα EMG και IMU δεδομένα που λαμβάνει.

- **IMU_EulerStar_Graph_node (imu_stars.py):** Είναι ο κόμβος στον οποίο γίνεται η μετατροπή των Quaternion μεταβλητών σε γωνίες Euler (υπενθύμιση-το Myo παράγει orientation data σε Quaternion μορφή) και κάνει publish σε δυο Topics. Το ένα είναι για την γραφική αναπαράσταση με τη βοήθεια συμβόλων “*” των τιμών των Roll , Pitch και Yaw και το άλλο με το πραγματικό value των παραπάνω 3ων μεταβλητών.
- **ARM_Limitations_node (arm_limits_setup.py) :** Ο κόμβος αυτός αρχικοποιεί τα όρια των κινήσεων του χρήστη που φοράει το Myo , δηλαδή όρια ανάτασης των χεριών του ή περιστροφής του καρπού του κτλ , έτσι ώστε να δημιουργήσει ένα profile για αυτόν και να καθορίσει τις κατάλληλες τιμές στο Robonaut ώστε να προσομοιώνει ακριβώς τις κινήσεις όπως τις πραγματοποιεί ο χρήστης, χωρίς καμιά διαφορά. Αυτό τα 'όρια' τα εξάγει σε ένα δικό του Topic, στο οποίο Topic κάνει Subscription το τελικό node, που καθορίζει την κίνηση του Robonaut.
- **Simulation_Controller_Menu_node (menu.py) :** Ο κόμβος αυτός είναι το βασικό χειριστήριο του simulation του χρήστη. Οι επιλογές που παρέχονται έχουν άμεση επίδραση στην όλη λειτουργία των κόμβων και των επεξεργασιών των δεδομένων του Myo. Συγκεκριμένα , οι επιλογές είναι,

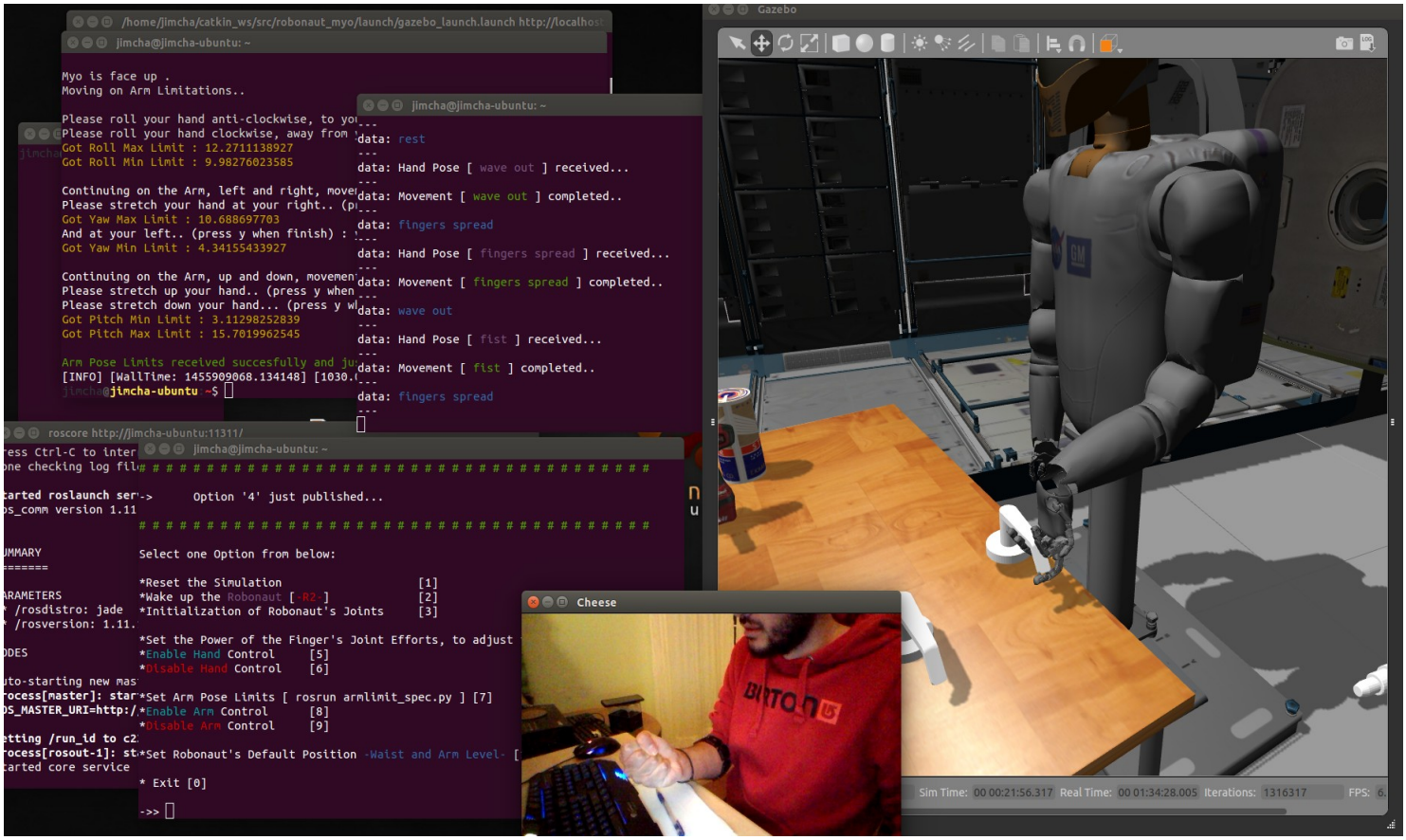


Πως φαίνεται, οι επιλογές έχουν να κάνουν κυρίως με τον κόσμο του Gazebo, με την έναρξή ή διακοπή του χειρισμού του ρομπότ (του χεριού του-του βραχίονα του ή και των δυο, αντίστοιχα) όπως και στην επιλογή του user αν θέλει να έχει μεγαλύτερο stiffness στις χειρονομίες του Robonaut ή να αλλάξει τη θέση του R2 στον χώρο. Οι επιλογές του εδώ, όπως και οι τιμές που ορίζει -για το stiffness π.χ- αποστέλλονται απευθείας στον "τελικό" κόμβο , οποί γίνεται όλη η επεξεργασία των δεδομένων του Myo και καθορίζονται οι τελικές εντολές για τα joints του R2.

- **Robonaut_Movement_Process_node** ή αλλιώς ο κεντρικός κόμβος. Στον κόμβο αυτό γίνεται η κατάλληλη επεξεργασία των raw data του Myo ανάλογα με τις επιλογές του χρήστη και την κατάσταση του R2. Το ROS κόμβος αυτός είναι και ο κόμβος ο οποίος θα καλέσει τα κατάλληλα ROS Services του GazeboSim για να αλληλεπιδράσει με τα joints του R2 και να μεταδώσει μια τυχόν κίνηση του βραχίωνα, κάποια χειρονομία ή ακόμα και να αρχικοποιήσει το World του Gazebo. Για το συγκεκριμένο node μπορείτε να δείτε περισσότερες μέσα στα σχόλια του κωδικά του *Robonaut_mainControl.py* , καθώς είναι αδύνατο να αναφέρω με λεπτομέρεια όλες τις δυνατές περιπτώσεις και υποπεριπτώσεις εδώ.

Συνοψίζοντας, αυτη είναι η συνολική περιγραφή του ROS package που δημιούργησα για το inteface του Myo με το προσομοιωμένο Robonaut 2. Σαφώς , αυτη η ιδέα μπορεί να εφαρμοστεί σε οποιοδήποτε προσομοιωμένο ρομποτικό σύστημα που προσομοιώνει κάποια κίνηση βραχίωνα, χεριού και όχι μόνο (π.χ drones και χειρισμός με το Myo), αλλά όπως επίσης μπορεί να εφαρμοστεί και σε πραγματικά ρομποτικά συστήματα, με τη βοήθεια των δυνατοτήτων του ROS.

Στην ιστοσελίδα υπάρχουν ανεβασμένα βίντεο που δείχνω τη διαδικασία ενεργοποίησης των ROS Nodes, καθώς και την εμφάνιση κατάλληλων ROS Topics, για να μπορούμε να δούμε τη μεταφορά των δεδομένων και τη δημοσίευση των μηνυμάτων του simulation, σε πραγματικό χρόνο. Επίσης μαζί με τα βίντεο, συμπεριλαμβάνω διάφορα screenshots κατά τη διάρκεια του Simulation όπως και RQT Graphs που απεικονίζουν τις σχέσεις μεταξύ των ROS Nodes , μέσω των ROS Topics , με τον τρόπο που περιγράφηκαν παραπάνω.



Instalation Instructions

Download and place the robonaut_myo package in your ROS workspace (like **~/catkin_ws/**). The package after the catkin_make built , will be placed in **~/catkin_ws/src/** .

$ cd ~/catkin_ws/

$ catkin_make

$ source devel/setup.bash

Running the ROS nodes (With a roscore running, run the below commands in separate terminals..)

$ rosrun gazebo_ros gazebo (gazebo_ros package must be installed)

Add the Robonaut2_st Model Inside the GazeboSim world. Close Gazebo, after doin this (the simulator runs in the background).

$ rosrun robonaut_myo classify_myoNode.py

For the first time press -y to pop-up the Pygame window to set up you hand gestures. After doing that close the Pygame window and rerun the command pressing -n this time to avoid the Graphic window (better solution for low specs Pcs).

$ rosrun robonaut_myo menu.py

To show up the Main Menu.

$ rosrun robonaut_myo arm_limits_setup.py

For the Arm limitation Setup.

And finally $ roslaunch robonaut_myo gazebo_launch.launch

To launch the Gazebo world -node- with the Robonaut in it.

(ROS Topic suggestions to Show up)

Do $ rostopic echo /movement_info

To see the Robonaut_Movement_Process Node Simulation Messages

Or $ rostopic echo /myo_imuEulerAngles_raw

To see the Myo's live IMU Euler Angles being broadcasting.

(Robonaut_myo package contains about 10 ROS Topics that publishes together, while the roslaunch is running. All these information that Topics publish, can be printed by using the command-line tool **rostopic**)