*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 - Senior Capstone Project

Software Engineering Focus

Final Deliverable

Project Title:

Robotic Arm 1.0

**Team Members:**

Curtis Cox

Shadeh Ferris-Francis

**Product Owner(s)**:

Nagarajan Prabakar

**Mentor(s)**:

Nagarajan Prabakar

**Instructor**: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2016 Discovery Lab*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*TeleBot is a telepresence, humanoid robot built for the purpose of assisting disabled police and military veterans in their return to the workforce by providing a means of remote law-enforcement surveillance.*

*The software solution discussed within this document is part of the third iteration of the TeleBot prototype. Existing and proposed features were developed to address usability, functionality, and testing limitations present in the previous versions of software in the robotic system.*

**Table of Contents**

**Introduction**

Current System

Purpose of New System

**User Stories**

Implemented User Stories

Pending User Stories

**Project Plan**

Hardware and Software Resources

Sprints Plan

*Sprint 1*

**System Design**

Architectural Patterns

System and Subsystem Decomposition

Deployment Diagram

Design Patterns

**System Validation**

**Glossary**

**Appendix**

Appendix A - UML Diagrams

*Static UML Diagrams*

*Dynamic UML Diagrams*

Appendix B - User Interface Design

Appendix C - Sprint Review Reports

Appendix D - Sprint Retrospective Reports

**References**

# Introduction

From conception in 2012, the TeleBot project of the Florida International University Discovery Lab has undergone three major software changes. The Spring 2016 Senior Design software solution presented in this document is part of the third iteration of the prototype software.

Discussion will be as follows: the current system will be briefly discussed, followed by the goal of the newly implemented system; a listing of the seven Sprints containing implemented user stories generated from requirements and respective diagrams will be covered, followed by an overview of the system design, software architecture, and design patterns in the software solution. System validation will conclude discussion within this document.

## Current System

In the previous version of the third iteration, the Operator would equip the YEI 3-Space Sensor wireless IMU sensors along her upper body. Head, shoulder, elbow, and wrist coordinate data would stream from sensors to the wireless dongle attached to the Operator PC. Running on the Operator PC would be the YEI 3-Space MoCap Studio program, which produced angular data for each joint. MoCap would be connected via TCP/IP to a Java-based Master/Publisher program, which abided by a publish-subscribe message-passing protocol.

This Master/Publisher program would use a mapping formula that took pre-set sensor maximum and minimum angles for a respective joint, pre-set servomotor maximum and minimum position units, and real-time angular data for this joint to produce a servomotor position. The Master/Publisher program would then publish or “write” the corresponding servomotor id, servomotor position, and servomotor speed to a “Topic”, or a message-mailbox in the publish-subscribe paradigm. This data would also be written to the console.

A Slave/Subscriber program residing on the TeleBot PC would then read this servomotor data from the Topic and process this data into a servomotor-readable form, and then transmit this data via serial communication to the CM-700 controller, which in turn processed this data in its firmware and transmitted the product to the daisy-chained servomotors which compose the Robot’s arms and head.

Compared to past software solutions that were centralized and relied on TCP/IP communication to transmit data, this system of this software solution was the first modular one of its kind, utilizing the publish-subscribe protocol of the Data Distribution Service. However, it ran primarily from the Eclipse IDE and relied on the console for observing servomotor-processed data. Port numbers were hard-coded, and on the Master/Publisher side, when connecting to MoCap Studio, the DDS publishing was also activated. Sensor maximum and minimum ranges for an individual Operator were hard-coded in a configuration file. Testing servomotor positions and ranges was dependent on a deprecated program, the Robotis Dynamixel Wizard, which had to be connected via a USB connector to the CM-700 controller board on the physical Robot, and feedback from the servomotors could only be seen in this program.

Also, the TeleBot system was severely limited for a solution whose purpose was to assist disabled police and military veterans in their return to the workforce. Use of the IMU sensors would require the ability to use the upper body in its entirety, including the left and right arms and hands. Mimicry of the Operator movement was somewhat hindered; robotic movement was limited in range and occasionally jittery.

## Purpose of New System

The software solution developed for Spring 2016 Semester addresses the limitations presented in the previous version of the third iteration. At the start of the semester, the old system was in a state that did not achieve control of Telebot’s arms. Therefore the first goal of the new system was to restore control to the system. When this was accomplished we moved on to improving the system. There are many features that we would like to add to Telebot and the new system implements some of those, including the voice control system. We worked to improve the reliability and efficiency of the Mocap Studio program. Finally we created some testing tools to help the Telebot Operator trouble shoot the system when problems arise.

# User Stories

From requirements collected in meetings with Dr. Prabakar, conversations with Discovery Lab members, daily Scrum meetings, and personal interactions with the system, user stories were formulated.

The following content briefly describes implemented user stories and pending user stories.

## Implemented User Stories

**User story #694 - Activate DDS Publishing when clicking Activate DDS Button - Optional**

As an Operator, I want to Activate IMU Master/Publisher Program DDS Publishing when I click Activate DDS Button so that activating communication to the DDS Topic is quick and not dependent on connecting to the Mocap System.

**User story #735 - Display Angular Data Received for Each Joint in the IMU Master/Publisher GUI - Required**

As an Operator, I want to see a display Angular Data Received for Each Joint in IMU Master/Publisher GUI so that I can monitor accuracy of the data received from the IMU sensors.

**User story #742 - Process Angular Data to Servomotor Units with IMU Master/Publisher Program - Required**

As an Operator, I want to Process Angular Data to Servomotor Units with IMU Master/Publisher Program so that the servomotors can process the data received.

**User story #743 - Connect Mocap System to IMU Master/Publisher Program - Required**

As an Operator, I want to Connect MoCap System to IMU Master/Publisher Program at a click of a button so that activating communication is quick and simple.

**User story #768 - Initiate Mocap Halt Rest Position - Optional**

As an Operator, I want when Mocap is stopped, robotic arms should go to rest position so that the Robot's servomotors are not strained from the weight of the arms when not in use.

**User story #769 - Process Commands to Servomotor Units with Voice Master/Publisher**

**Program - Required**

As an Operator, I want to Process commands to Servomotor Units with Voice Master/Publisher Program so that a command can move the robotic arms to rest, stop, and shield positions.

**User story #770 - Connect Voice Control System to Voice Master/Publisher Program - Required**

As an Operator, I want to Connect Voice Control System to Voice Master/Publisher Program with the click of a button so that activating communication is quick and simple.

**User story #789 - Understand DDS Message delivery - Optional**

As a robot programmer I would like to understand how DDS messages are received by telebot so that I can identify the reasons for delay in the system.

**User Story #796 - Arm Servo Test Driver - Required**

As a robot operator I would like to have a servo test driver so that I can ensure that the servos are functioning properly

**User Story #808 - Monitor Servo Feedback - Optional**

As a Telebot operator I would like to be able to get feedback from the servos so that I can detect warning signs of servo trouble.

**User Story #816 - DDS servo correction - Required**

As a Telebot Operator I would like the DDS program to correct the servo data from Mocap studios so that Telebot’s arm move smoothly.

**User Story #839 - Servo Feedback Latency - Optional**

As a Telebot Operator I would like the servo feedback GUI to display feedback as soon as it is received so that I can know when the servos reach their position.

**User Story #840 - Mocap algorithm improvement - Optional**

As a Telebot Operator I would like to have Mocap Studio send angle data for the head as an offset from the rest value so that the head movements look more natural.

**User Story #851 - Arm Quaternions - Required**

As a Telebot Operator I would like Mocap studio to use quaternion math to find the angles for the joints in Telebot’s arms so that the angles will be more accurate.

## Pending User Stories

**User Story # 664 - Only Real-Time Data Sent from Operator to Robot - Required**

As an Operator, I want only real-time, up-to-date data to be sent from the Operator to the Robot so that the Robot is accurately mimicking the Operator's movements.

**User Story # 665 - Send Data from the Operator to the Robot with Publish/Subscriber Programs - Required**

As a Operator, I want to send data from the Operator to the Robot with publish/subscribe Master-Slave programs so that the Operator can easily connect and disconnect to the Robot.

**User Story # 666 - Activate DDS Publishing when connecting Kinect Control System to Kinect Master/Publisher Program - Optional**

As an Operator, I want to activate the Kinect Master/Publisher Program DDS Publishing when I connect to the Kinect Control System so that activating communication is quick and simple.

**User Story # 667 - Process Operator Data with Kinect Control System - Required**

As an Operator, I want to Process Operator Data with Kinect Control System so that the Master/Publisher program can receive and process this data without too many calculations needed.

**User Story # 668 - Display Angular Data Received for Each Joint in Kinect Master/Publisher GUI - Optional**

As an Operator, I want to see a display of Angular Data Received for Each Joint in Kinect Master/Publisher GUI so that the Operator can monitor the accuracy of the data.

**User Story # 671 - Process Angular Data to Servomotor Units with Kinect Master/Publisher Program - Required**

As an Operator, I want to Process Angular Data to Servomotor Units with Kinect Master/Publisher Program so that the servomotors can process the data received.

**User Story # 687 - Transmit Data to Servomotors with Slave/Subscriber Program - Required**

As an Operator, I want to transmit Data to Servomotors with Slave/Subscriber Program so that the Robot can move in real-time according to the data streaming in from the Operator

**User Story # 693 - Connect Kinect Control System to Kinect Master/Publisher Program - Required**

As an Operator, I want to connect the Kinect Control System to Kinect Master/Publisher Program with the click of a button so that activating communication is quick and simple.

**User Story # 730 - Display Positions Received for Each Servomotor in the Slave/Subscriber Program GUI - Optional**

As an Operator, I want to see a display of Positions Received for Each Servomotor in the Slave/Subscriber Program GUI so that I can monitor accuracy of positional data

**User Story # 767 - Activate DDS Subscribing when Launching Slave/Subscriber Program - Optional**

As an Operator, I want to activate Slave/Subscriber Program DDS Subscribing upon launch of the Slave/Subscriber Program so that I can easily connect and disconnect to the Robot

**User Story # 772 - Activate DDS Publishing - Optional**

As an Operator, I want to Activate Voice Master/Publisher Program DDS Publishing when I connect to the Voice Master/Publisher Program so that activating communication is quick and simple.

**User Story # 773 -Collect Arm Joint Ranges for Operator - Optional**

An an Operator, I want the creation of a program that collects the arm joint ranges for any operator so that each Operator that uses the Robot will have accurate arm-ranges mapped to the Robot.

**User Story # 774 - Switch from IMU Sensors to Kinect Sensor to Voice in that Priority - Optional**

As an Operator, I want the ability to switch from IMU sensors to Kinect Sensor to Voice (priority in that order from high to low) so that activating and switching communication when needed is quick and simple.

**User Story # 775 - Display Operator Skeleton Frame - Required**

As an Operator, I want to see a display of the Operator Skeleton Frame so that the Operator can monitor the accuracy of the Kinect device.

**User Story # 807 - DDS Master to Slave test tool - Optional**

As a Telebot operator I would like to have a tool to test the DDS path from the Master Program to the Slave program.

**User Story # 850 - DDS Longevity - Optional**

As a Telebot Operator I would like to ensure that the DDS system does not shut down until requested so that operation is not interrupted.

# Project Plan

The TeleBot system required numerous hardware and software tools for development, operation, and maintenance. These tools were selected based on their low-cost, availability, ease of use and/or reliability.

## Hardware and Software Resources

|  |
| --- |
| Eclipse IDE |
| GitHub |
| YEI Technology 3-Space IMU Wireless Sensors and Dongle |
| YEI Technology 3-Space MoCap Studio |
| YEI Technology 3-Space Suite |
| RTI Data Distribution Service |
| Robotis Dynamixel Robot Actuator [Servo] MX-106R and MX-64T |
| CM700 Robotis Servo Controller and SUB Board |
| Robotis RoboPlus Software |
| Ubuntu Linux 14.04 |
| Git |
| Java 1.7+ |
| Python 2.7 |
| DLink Router |
| BitVoicer |
| Free TTS |
| Java Simple Serial Connector Library |
| Microsoft Windows 7 and 8 |
| Simple Logging Facade for Java API |
| Robotis USB2Dynamixel Adapter |
| ASUS |
| MacBook Pro |
| ASRock |
| ASRock |
| Linksys Router |

## Sprints Plan

### Sprint 1

(01/16/2016 - 01/29/2016)

**User story #789 - Understand DDS Message delivery**

As a robot programmer I would like to understand how DDS messages are received by telebot so that I can identify the reasons for delay in the system.

***Tasks***

* 855 Confirm DDS shuts down
* 867 Document DDS

### Acceptance Criteria

* Cause of message delay has been Identified

***Modeling***

* Appendix A Figure #8 - User Story #789 Class Diagram
* Appendix A Figure #29 - User Story #789 Use Case Diagram
* Appendix A Figure #30 - User Story #789 Publisher Sequence Diagram
* Appendix A Figure #31 - User Story #789 Subscriber Sequence Diagram

### Sprint 2

(01/30/2016 - 02/12/2016)

**User Story #796 - Arm Servo Test Driver**

As a robot operator I would like to have a servo test driver so that I can ensure that the servos are functioning properly

***Tasks***

* 797 Create Servo Catalog
* 798 Create Servo Driver Functions
* 799 Create Servo Drive User Interface

### Acceptance Criteria

* Test Driver can effectively control each of the servos in the robot’s arms.

***Modeling***

* Appendix A Figure #9 - User Story #796 Class Diagram
* Appendix A Figure #32 - User Story #796 Use Case Diagram
* Appendix A Figure #33 - User Story #796 Sequence Diagram

### Sprint 3

(02/13/2016 - 02/26/2016)

User Story #800 - Create a Windows Virtual Machine to Linux Host Machine Communication Protocol **-** not completed

**User Story #808 - Monitor Servo Feedback**

As a Telebot operator I would like to be able to get feedback from the servos so that I can detect warning signs of servo trouble.

***Tasks***

* 809 Check CM-700 for two way communication
* 810 Create Servo Monitor class
* 815 Create Servo Feedback Test GUI

### Acceptance Criteria

* Test program can receive feedback from servos
* Program can determine if the actual position of the servo matches the requested position.

***Modeling***

* Appendix A Figure #10 - User Story #808 Class Diagram
* Appendix A Figure #34 - User Story #808 Use Case Diagram
* Appendix A Figure #35 - User Story #808 Sequence Diagram

### Sprint 4

(02/27/2016 - 03/11/2016)

**User Story #742 - Process Angular Data to Servomotor Units with IMU Master/Publisher Program**

As an Operator, I want to Process Angular Data to Servomotor Units with IMU Master/Publisher Program so that the servomotors can process the data received.

***Tasks***

* 812 Create Use Case
* 813 Create Sequence Diagram
* 811 Create Class Diagram
* 821 Process Angular Data to Servomotor Units
* 814 Create Test Cases
* 830 Create User Guide

### Acceptance Criteria

* Given that the IMU Master/Publisher Program is collecting data from the Mocap System, when the angular data is received, the IMU Master/Publisher program should process this data to servomotor units with a mapping formula.

***Modeling***

* Appendix A Figure # - User Story #742 Class Diagram
* Appendix A Figure # - User Story #742 Use Case Diagram
* Appendix A Figure # - User Story #742 Sequence Diagram

**User Story #735 - Display Angular Data Received for Each Joint in the IMU Master/Publisher GUI**

As an Operator, I want to see a display Angular Data Received for Each Joint in IMU Master/Publisher GUI so that I can monitor accuracy of the data received from the IMU sensors.

***Tasks***

* 809 Check CM-700 for two way communication
* 810 Create Servo Monitor class
* 815 Create Servo Feedback Test GUI

### Acceptance Criteria

* Given that the Mocap System and the IMU Master/Publisher are connected, when angular data is received by the IMU Master/Publisher, the IMU Master/Publisher GUI should display this data for each joint.

***Modeling***

* Appendix A Figure # - User Story #735 Class Diagram
* Appendix A Figure # - User Story #735 Use Case Diagram
* Appendix A Figure # - User Story #735 Sequence Diagram

**User Story #816 - DDS servo correction**

As a Telebot Operator I would like the DDS program to correct the servo data from Mocap studios so that Telebot’s arm move smoothly.

***Tasks***

* 822 Learn Status of story
* 823 Merge github(s)
* 824 Identify error prone movements
* 825 Confirm noninterference with other programs
* 828 Reprogram Hardware Memory
* 831 Move correction to Mocap

### Acceptance Criteria

* Telebot’s arms correctly mimic the user’s movements.
* Telebot’s arms move smoothly from one position to the next.

***Modeling***

* Appendix A Figure #11 - User Story #816 Class Diagram
* Appendix A Figure #36 - User Story #816 Use Case Diagram
* Appendix A Figure #37 - User Story #816 Mocap Studio Sequence Diagram
* Appendix A Figure #38 - User Story #816 IMU Master/Publisher Sequence Diagram

### Sprint 5

(03/21/2016 - 04/01/2016)

**User Story #694 - Activate DDS Publishing when clicking Activate DDS Button**

As an Operator, I want to Activate IMU Master/Publisher Program DDS Publishing when I click Activate DDS Button so that activating communication to the DDS Topic is quick and not dependent on connecting to the Mocap System.

***Tasks***

* 703 Create Use Case
* 704 Create Use Case Diagram
* 705 Create Sequence Diagram
* 706 Create Class Diagram
* 707 Creation of DDS Activation Button
* 741 Create Test Cases
* 845 Create User Guide

### Acceptance Criteria

* Given that whether or not the Mocap System is launched, when the Activate DDS Button is clicked in the IMU Master/Publisher, DDS Publishing should be activated.

***Modeling***

* Appendix A Figure # - User Story #694 Class Diagram
* Appendix A Figure # - User Story #694 Use Case Diagram
* Appendix A Figure # - User Story #694 Sequence Diagram

**User Story #743 - Connect Mocap System to IMU Master/Publisher Program**

As an Operator, I want to Connect MoCap System to IMU Master/Publisher Program at a click of a button so that activating communication is quick and simple.

***Tasks***

* 832 Create Use Case
* 833 Create Use Case Diagram
* 834 Create Sequence Diagram
* 835 Create Class Diagram
* 836 Create Connection Feature
* 837 Create Test Cases
* 838 Create User Guide

### Acceptance Criteria

* Given that Mocap Studio and the IMU Master/Publisher Program are on the same computer system and the IMU Master/Publisher is active, when the local address and set port are entered in Mocap Studio GUI and 'Connect' is clicked, Mocap Studio should connect to the IMU Master/Publisher Program.

***Modeling***

* Appendix A Figure # - User Story #743 Class Diagram
* Appendix A Figure # - User Story #743 Use Case Diagram
* Appendix A Figure # - User Story #743 Sequence Diagram

**User Story #840 - Mocap algorithm improvement**

As a Telebot Operator I would like to have Mocap Studio send angle data for the head as an offset from the rest value so that the head movements look more natural.

***Tasks***

* 841 Correct Head Yaw- Animator
* 842 Head Yaw Correction - DDS
* 843 Head Tilt Correction - Animator
* 844 Head Tilt Correction - DDS

### Acceptance Criteria

* Telebot’s head should look forward when not turned.
* Telebot’s chin should be raised to a natural position when not looking up or down.

***Modeling***

* Appendix A Figure #13 - User Story #840 Class Diagram
* Appendix A Figure #42 - User Story #840 Use Case Diagram
* Appendix A Figure #43 - User Story #840 Mocap Studio Sequence Diagram
* Appendix A Figure #44 - User Story #840 IMU Master/Publisher Sequence Diagram

**User Story #839 - Servo Feedback Latency**

### As a Telebot Operator I would like the servo feedback GUI to display feedback as soon as it is received so that I can know when the servos reach their position.

***Tasks***

* 846 Change JLables to JTextFields
* 847 Regression Test
* 848 Latency Test
* 849 Add Head Rest

### Acceptance Criteria

* The servo feedback GUI should refresh as soon as it receives new data.

***Modeling***

* Appendix A Figure #12 - User Story #839 Class Diagram
* Appendix A Figure #39 - User Story #839 Use Case Diagram
* Appendix A Figure #40 - User Story #839 Timer Sequence Diagram
* Appendix A Figure #41 - User Story #839 Main Sequence Diagram

### Sprint 6

(04/02/2016 - 04/15/2016)

User Story #789 not completed

**User Story #768 - Initiate Mocap Halt Rest Position**

### As an Operator, I want when Mocap is stopped, robotic arms should go to rest position so that the Robot's servomotors are not strained from the weight of the arms when not in use.

***Tasks***

* 857 Create Use Case
* 858 Create Use Case Diagram
* 859 Create Sequence Diagram
* 860 Create Class Diagram
* 861 Develop Rest Position Feature
* 862 Create Test Cases
* 863 Create User Guide

### Acceptance Criteria

* Given that the Mocap System, IMU Master/Publisher Program and Slave/Subscriber Program are in operation, when the Mocap System is terminated, the servomotors on the Robot Side should assume rest position.

***Modeling***

* Appendix A Figure # - User Story #768 Class Diagram
* Appendix A Figure # - User Story #768 Use Case Diagram
* Appendix A Figure # - User Story #768 Mocap Studio Sequence Diagram
* Appendix A Figure # - User Story #768 IMU Master/Publisher Sequence Diagram

**User Story #770 - Connect Voice Control System to Voice Master/Publisher Program**

### As an Operator, I want to Connect Voice Control System to Voice Master/Publisher Program with the click of a button so that activating communication is quick and simple.

***Tasks***

* 778 Create Use Case
* 784 Create Sequence Diagram
* 864 Create Class Diagram
* 785 Create/Develop/Configure Connection Capabilities for Voice Control System
* 786 Create Connection Capabilities for Voice Master/Publisher Program
* 781 Create Test Case
* 865 Create User Guide

### Acceptance Criteria

* Given that the Voice Control System and the Voice Master/Publisher Program are on the same computer system and the Voice Master/Publisher is active, when the local address and set port are entered in the Voice Control System GUI and 'Connect' is clicked, the Voice Control System should connect to the Voice Master/Publisher Program.

***Modeling***

* Appendix A Figure # - User Story #770 Class Diagram
* Appendix A Figure # - User Story #770 Use Case Diagram
* Appendix A Figure # - User Story #770 Mocap Studio Sequence Diagram
* Appendix A Figure # - User Story #770 IMU Master/Publisher Sequence Diagram

**User Story #851 - Arm Quaternions**

### As a Telebot Operator I would like Mocap studio to use quaternion math to find the angles for the joints in Telebot’s arms so that the angles will be more accurate.

***Tasks***

* 852 Identify Euler Angles
* 853 Update calculateJointAngles
* 854 Update onTimeUpdate
* 866 Update DDS Publisher

### Acceptance Criteria

* Telebot’s arm’s accurately mimic the Telebot Operator’s arm movements

***Modeling***

* Appendix A Figure #14 - User Story #851 Class Diagram
* Appendix A Figure #45 - User Story #851 Use Case Diagram
* Appendix A Figure #46 - User Story #851 Mocap Studio Sequence Diagram
* Appendix A Figure #47 - User Story #851 IMU Master/Publisher Sequence Diagram

### Sprint 7

(04/16/2015 - 04/29/2016)

**User story #789 - Understand DDS Message delivery**

As an Operator, I want to Process commands to Servomotor Units with Voice Master/Publisher Program so that a command can move the robotic arms to rest, stop, and shield positions.

***Tasks***

* Create Use Case
* Create Use Case Diagram
* Create Sequence Diagram
* Create Class Diagram
* Develop command processing
* Create Test Cases
* Create User Guide

### Acceptance Criteria

* Given that the Voice Master/Publisher Program is collecting string command data from the Voice Control System, when the string command data is received, the Voice Master/Publisher program should process this string command data to the appropriate assigned servomotor units.

***Modeling***

* Appendix A Figure # - User Story #769 Class Diagram
* Appendix A Figure # - User Story #769 Use Case Diagram
* Appendix A Figure # - User Story #769 Publisher Sequence Diagram
* Appendix A Figure # - User Story #769 Subscriber Sequence Diagram

**User story #789 - Understand DDS Message delivery**

As a robot programmer I would like to understand how DDS messages are received by telebot so that I can identify the reasons for delay in the system.

***Tasks***

* 855 Confirm DDS shuts down
* 867 Document DDS

### Acceptance Criteria

* Cause of message delay has been Identified

***Modeling***

* Appendix A Figure #8 - User Story #789 Class Diagram
* Appendix A Figure #29 - User Story #789 Use Case Diagram
* Appendix A Figure #30 - User Story #789 Publisher Sequence Diagram
* Appendix A Figure #31 - User Story #789 Subscriber Sequence Diagram

# System Design

For the system design for the TeleBot software solution, several architectural and design patterns were utilized; also, the Operator PC, Robot PC, and respective devices are contained in the Deployment Diagram.

## Architectural Patterns

Pipe and Filter - As messages move from their origin, through DDS, to Telebot and the CM-700 servo controller, they are modified a little bit at each step to prepare them to be sent to the servos.

Event Driven - Most of the systems are idle waiting for an event to trigger an action.

Model-View-Controller - The Servo Test App presents a view to the Telebot Operator. It maintains a model of how the servos should be set. When reality differs from the model the operator is alerted to a problem.

## System and Subsystem Decomposition

Mocap Studio is the motion capture software and BitVoicer is the voice recognition software. They are both commercial software products that use Master/Publisher adapter programs to publish to DDS. Slave Subscriber converts DDS messages to serial messages to send to the CM-700 Servo Controller. Telebot Arm Servo Feedback Tester App and CM-700 is the only link with two way communication, as it is separate from the main pipe and filter architecture of the main system.

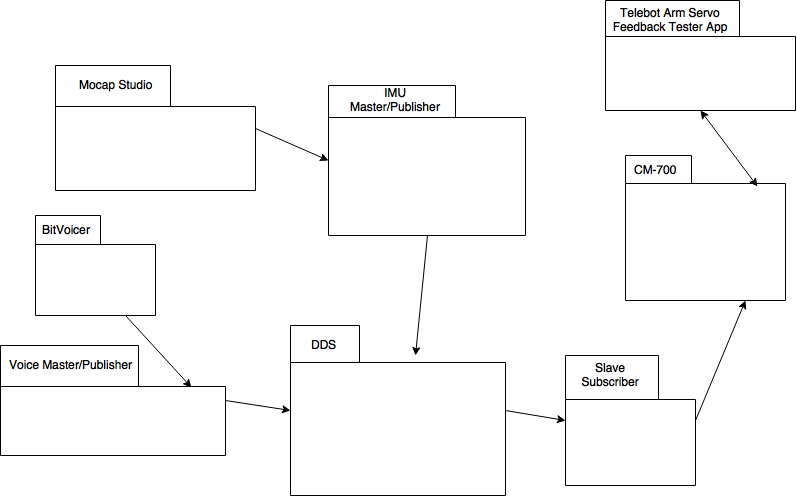


Figure #1 - Package Diagram

## Deployment Diagram

Deployment Diagram.png

Figure #2 - Deployment Diagram

## Design Patterns

Singleton - Used to ensure only one instance of a model class is created.

Publish/Subscribe - This allows us to have several control programs simultaneously controlling Telebot.

Adapter - Used to let commercial software publish to DDS.

# System Validation

System validation was performed by deploying the software on Telebot and/or the remote computer in the Discovery Lab. System tests were devised to ensure that the new software functioned and interacted with other subsystems appropriately in the live environment. Subsystem tests were performed using JUnit for code written in Java. The Mocap Studios Python code was tested using driver functions and stubs when necessary.

**User story #694 - Activate DDS Publishing when clicking Activate DDS Button**

System Tests

* Test Case ID: 694-I001-SunnyDay - Purpose: When the Publish button is pressed, the DDS Publishing and Data Writing should be activated.
* Test Case ID: 694-I002-RainyDay - Purpose: When Mocap and the IMU Master/Publisher are not connected, the DDS Publishing and Data Writing should not be activated.
* Test Case ID: 694-I003-RainyDay - Purpose: When the sensors are not connected, the DDS Publishing should not have anything written in the console/posted to the Topic.

Subsystem Tests

* Test Case ID: 694-U001-TelebotMasterArmsTCPComponent-InitiateTransmissionTest - Purpose: Test if the DDS Publishing is activated in the model.
* Test Case ID: 694-U001-TelebotMasterArmsTCPComponent-InitiateDataWriter - Purpose: Test if the DDS Data Writing is activated.

**User story #735 - Display Angular Data Received for Each Joint in the IMU Master/Publisher GUI**

System Tests

* Test Case ID: 735-I001-SunnyDay - Purpose: Test that joint data is written in the appropriate joint slot in the GUI.
* Test Case ID: 735-I001-RainyDay - Purpose: Test that joint data is not updated or erroneous data is not written in the GUI at the point when Mocap System and IMU Master/Publisher are disconnected.
* Test Case ID: 735-I001-RainyDay - Purpose: Test that erroneous data is not being written to the GUI when the Mocap System and the IMU Master/Publisher are connected and the sensors are not turned on.

Subsystem Tests

* Test Case ID: 735-U001-TelebotMasterArmsTCPController-ChangeText - Purpose: Test that joint data is written in the appropriate joint slot in the GUI.

**User story #742 - Process Angular Data to Servomotor Units with IMU Master/Publisher Program**

System Tests

* Test Case ID: 742-I001-TelebotMasterArmsTCP-SunnyDay - Purpose: Test that streaming angular data from the Mocap System can be processed into servo positions by the IMU Master/Publisher.
* Test Case ID: 742-I003-TelebotMasterArmsTCP-RainyDay - Purpose: Test that when null angular data from Mocap Studio is received,no output is processed by the IMU Master/Publisher.

Subsystem Tests

* Test Case ID: 742-U001-ServoDataMapperTest-Constrain - Purpose: Test that a servo position value is constrained between a maximum and minimum possible servo positions values.
* Test Case ID: 742-U002-ServoDataMapperTest-Map - Purpose: Test that sensor values received are mapped into servo position values based on a maximum and minimum of a range of possible sensor values and possible servo values.
* Test Case ID: 742-U003-ServoDataMapperTest-Process - Purpose: Test that sensor values received are mapped into servo position values based on a maximum and minimum of a range of possible sensor values and possible servo values and constrain this value between a maximum and minimum of servo value range.
* Test Case ID: 742-U004-YEIDataParser-Parse - Purpose: Test that sensor data received can be parsed and components of this data can be assigned to the jointType, x, y, and z attributes of a YEIDataModel instance representing this data.
* Test Case ID: 742-U005-TelebotMasterArmsTCPComponentTest-Callback - Purpose: Test that data is received and processed.
* Test Case ID: 742-U006-TelebotMasterArmsTCPComponentTest-generatePositions - Purpose: Test that for a specific joint, two servo positions are generated and only updated positions are written.
* Test Case ID: 742-U007-TelebotMasterArmsTCPComponentTest-writeServoData - Purpose: Test if the data processed can be written to the Object whose data is published to the DDS Topic.

**User story #743 - Connect Mocap System to IMU Master/Publisher Program**

System Tests

* Test Case ID: 743-I001 - Purpose: When the port number is entered and the Listen button is clicked, the IMU Master/Publisher should listen to that port.

Subsystem Tests

* Test Case ID: 743-U001-TelebotMasterArmsTCPComponent-Initiate - Purpose: When the port number is entered and the Listen button is clicked, the server socket should be initiated.

**User story #768 - Initiate Mocap Halt Rest Position**

System Tests

* Test Case ID: 768-I001-SunnyDay - Purpose: When the Mocap System is disconnected from the IMU Master/Publisher, the servomotor rest positions should be written to the DDS Topic.
* Test Case ID: 768-I002-RainyDay - Purpose: When the Mocap System is disconnected from the IMU Master/Publisher, the servomotor rest positions should be written to the DDS Topic only once.
* Test Case ID: 768-I003-RainyDay - Purpose: When the Mocap System is connected to the IMU Master/Publisher, the servomotor rest positions should not be written to the DDS Topic.

Subsystem Tests

* Test Case ID: 768-U001-TelebotMasterArmsTCPComponent-GeneratePositions - Purpose: When the “null” data is received when Mocap System is disconnected from the IMU Master/Publisher, the servomotor rest positions should be written to the DDS Topic.

**User story #769 - Process Commands to Servomotor Units with Voice Master/Publisher Program**

System Tests

* Test Case ID: 770-I001-SunnyDay - Purpose: When the Operator says the voice commands “Stop”, “Shield” or “Rest”, the Robot’s arms should adjust to the appropriate position corresponding to that command.
* Test Case ID: 770-I002-RainyDay - Purpose: When the Operator says a phrase other than a voice command, no data should be processed.

Subsystem Tests

* Test Case ID: 769-U001-TelebotMasterArmsTCPVoiceComponent- GeneratePositionsTest - Purpose: With a command given, the respective servomotor positions should be processed.

**User story #770 - Connect Voice Control System to Voice Master/Publisher Program**

System Tests

* Test Case ID: 770-I001-SunnyDay - Purpose: When the port number is entered and the Listen button is clicked, the Voice Control Master/Publisher should listen to that port.
* Test Case ID: 770-I002-RainyDay - Purpose: When the port number has not been entered and the Listen button is clicked, a message asking the user to enter a valid port number should be displayed.

Subsystem Tests

* Test Case ID: 770-U001-TelebotMasterArmsTCPVoiceComponent-Initiate - Purpose: With the port number, the server socket should be initiated.

**User Story #796 - Arm Servo Test Driver**

System Tests

* Test Case ID: Task799-001 ServoTestApp test on Telebot - Purpose: Test that the ServoTestApp will correctly control the servos in Telebot’s arms.

Subsystem Tests

* Test Case ID: Task798-001 ServoControl constructor - Purpose: Test that the ServoControl class singleton is created when getSingleton() is called and that the serial connection is opened.
* Test Case ID: Task798-002 PositionsModel class constructor - Purpose: Test that the PositionsModel class singleton is created when getSingleton() is called.
* Test Case ID: Task798-003 ServoControl newValue method - valid value - Purpose: Test that the ServoControl class newValue method will update the model to reflect the new value for the selected servo when a value in the valid range for that servo is selected.
* Test Case ID: Task798-004 ServoControl newValue method - value too high - Purpose: Test that the ServoControl class newValue method will update the model to reflect the maximum value for the selected servo when a value higher than the valid range for that servo is selected.
* Test Case ID: Task798-005 ServoControl newValue method - value too low - Purpose: Test that the ServoControl class newValue method will update the model to reflect the minimum value for the selected servo when a value lower than the valid range for that servo is selected.
* Test Case ID: Task798-006 PositionsModel getPosition method - Purpose: Test that the PositionsModel class getPosition method returns the value stored for the selected servo.
* Test Case ID: Task798-007 ServoControl newValue method - other servo not at rest - Purpose: Test that the ServoControl class newValue method will set the other servo in the same arm as the selected servo to their rest position before setting the selected servo to the new value.

**User Story #808 - Monitor Servo Feedback**

System Tests

* Test Case ID: Task808-005 Servo Control of Telebot - Purpose: Test that the Servo Feedback Test App will correctly control Telebot’s servos and correctly read feedback from the servos.
* Test Case ID: Task808-006 Firmware backward compatible - Purpose: Test that the new CM-700 firmware is compatible with old software versions.

Subsystem Tests

* Test Case ID: Task810-001 ServoFeedback constructor - Purpose: Test that the ServoFeedback class singleton is created when getSingleton() is called and only one instance of the class is created after multiple calls to getSingleton().
* Test Case ID: Task810-002 serialEvent - Purpose: Test that the ServoFeedback class responds to notification of available serial data by reading and parsing the data.
* Test Case ID: Task810-003 Serial message parsing - Purpose: Test that the ServoFeedback class responds to notification of available serial data by reading and parsing the data.
* Test Case ID: Task815-004 GUI Feedback Labels - Purpose: Test that the ServoFeedbackTestGui class responds to the refreshView method by updating the text of the appropriate label with the current and requested positions of the selected servo and changing the text color of the label as appropriate.

**User Story #816 - DDS servo correction**

System Tests

* Test Case ID: Task816-005 Test New system on Telebot - Purpose: Ensure that Telebot’s arms and head will mimic the movements of the Telebot Operator when using the new algorithm.

Subsystem Tests

* Test Case ID: Task816-001 Valid Left Elbow mapping - Purpose: Ensure that the valid angle values for the left elbow from Mocap Studio will be translated to correct servo positions for servo 23.
* Test Case ID: Task816-002 Invalid Left Elbow mapping - Purpose: Ensure that the invalid angle values for the left elbow from Mocap Studio will be translated to correct servo positions for servo 23.
* Test Case ID: Task816-003 Valid Right Shoulder mapping - Purpose: Ensure that the valid angle values for the right shoulder from Mocap Studio will be translated to correct servo positions for servo 31.
* Test Case ID: Task816-004 Invalid Right Shoulder mapping - Purpose: Ensure that the invalid angle values for the right shoulder from Mocap Studio will be translated to correct servo positions for servo 31.

**User Story #839 - Servo Feedback Latency**

System Tests

* Test Case ID: Task839-005 Servo Control of Telebot - Purpose: Test that the Servo Feedback Test App will correctly control Telebot’s servos and correctly read feedback from the servos.

Subsystem Tests

* Test Case ID: Task839-Regress001 ServoFeedback constructor - Purpose: Test that the ServoFeedback class singleton is created when getSingleton() is called and only one instance of the class is created after multiple calls to getSingleton().
* Test Case ID: Task839-Regress002 serialEvent - Purpose: Test that the ServoFeedback class responds to notification of available serial data by reading and parsing the data.
* Test Case ID: Task839-Regress003 Serial message parsing - Purpose: Test that the ServoFeedback class responds to notification of available serial data by reading and parsing the data.
* Test Case ID: Task839-004 GUI Feedback Labels - Purpose: Test that the ServoFeedbackTestGui class responds to the refreshView method by updating the text of the appropriate label with the current and requested positions of the selected servo and changing the text color of the label as appropriate.

**User Story #840 - Mocap algorithm improvement**

System Tests

* Test Case ID: Task840-005 Test New system on Telebot - Purpose: Ensure that Telebot’s head will mimic the movements of the Telebot Operator when using the new algorithm.

Subsystem Tests

* Test Case ID: Task840-001 Valid head yaw mapping - Purpose: Ensure that the valid angle values for the head from Mocap Studio will be translated to correct servo positions for servo 11.
* Test Case ID: Task840-002 Invalid head yaw mapping - Purpose: Ensure that the invalid angle values for the head from Mocap Studio will be translated to correct servo positions for servo 11.
* Test Case ID: Task840-003 Valid head tilt mapping - Purpose: Ensure that the valid angle values for the head from Mocap Studio will be translated to correct servo positions for servo 10.
* Test Case ID: Task840-004 Invalid head tilt mapping - Purpose: Ensure that the invalid angle values for the head from Mocap Studio will be translated to correct servo positions for servo 10.

**User Story #851 - Arm Quaternions**

System Tests

* Test Case ID: Task851-005 Test New system on Telebot - Purpose: Ensure that Telebot’s arms will mimic the movements of the Telebot Operator when using the new algorithm.

Subsystem Tests

* Test Case ID: Task851-001 Valid left elbow roll mapping - Purpose: Ensure that the valid angle values for the left elbow from Mocap Studio will be translated to correct servo positions for servo 23.
* Test Case ID: Task851-002 Invalid left elbow roll mapping - Purpose: Ensure that the invalid angle values for the left elbow from Mocap Studio will be translated to correct servo positions for servo 23.
* Test Case ID: Task851-003 Valid left arm yaw mapping - Purpose: Ensure that the valid angle values for the left arm from Mocap Studio will be translated to correct servo positions for servo 22.
* Test Case ID: Task851-004 Invalid left arm yaw mapping - Purpose: Ensure that the invalid angle values for the left arm from Mocap Studio will be translated to correct servo positions for servo 22.

# Glossary

· DDS - Data Distribution Service - A many to many, publish / subscribe communication service.

· IMU - Inertial Measurement Unit - An electronic device that measures acceleration. The acceleration data can be used to compute the IMU’s orientation in 3D space.

· Servomotor - A type of motor that provides precise control over the number of degrees of rotation. Servomotors are frequently used to actuate the joints of robots because of this precision of control.

· Master/Publisher - A program that produces messages to be disseminated to other programs through DDS. Due to the event driven nature of the system, we refer to publishers as masters, because they instigate work in other programs when they publish messages.

· Slave/Subscriber - A program that consumes messages posted to a topic on DDS. Due to the event driven nature of the system, we refer to subscribers as slaves because they are coerced into work when a message is posted to their topic.

# Appendix

## Appendix A - UML Diagrams

### Static UML Diagrams

694CD.png

Figure #1 - User Story #694 Class Diagram

735CD.png

Figure #2 - User Story #735 Class Diagram

CD Process Angular Data.png

Figure #3 - User Story #742 Class Diagram

743CD.png

Figure #4 - User Story #743 Class Diagram

768CD.png

Figure #5 - User Story #768 Class Diagram

769CD.png

Figure #6 - User Story #769 Class Diagram

770CD.png

Figure #7 - User Story #770 Class Diagram

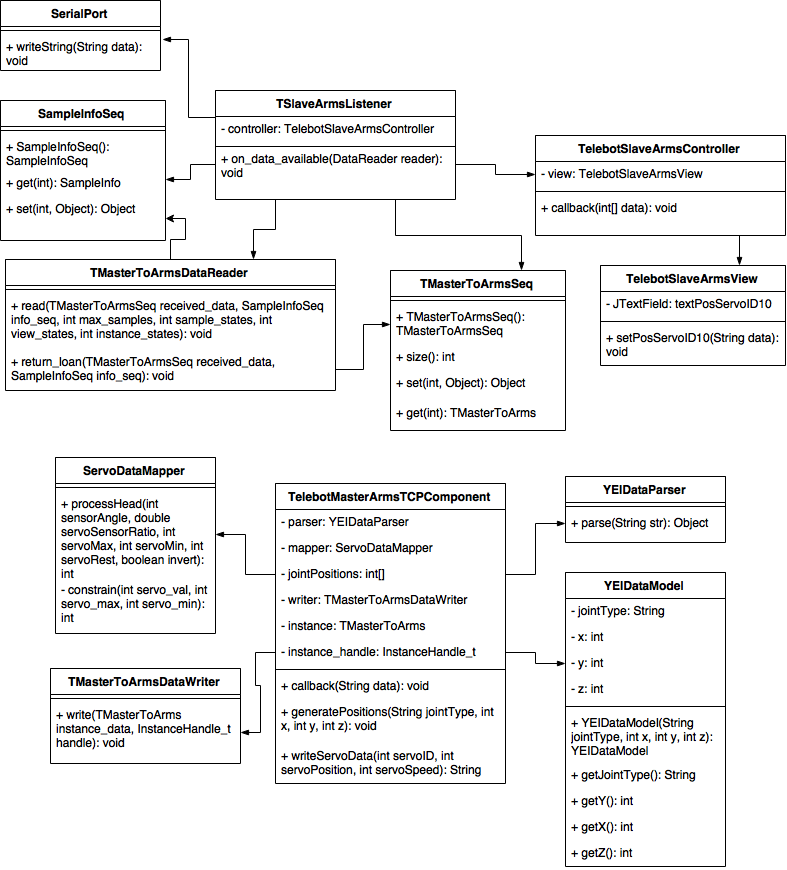


Figure #8 - User Story #789 Class Diagram

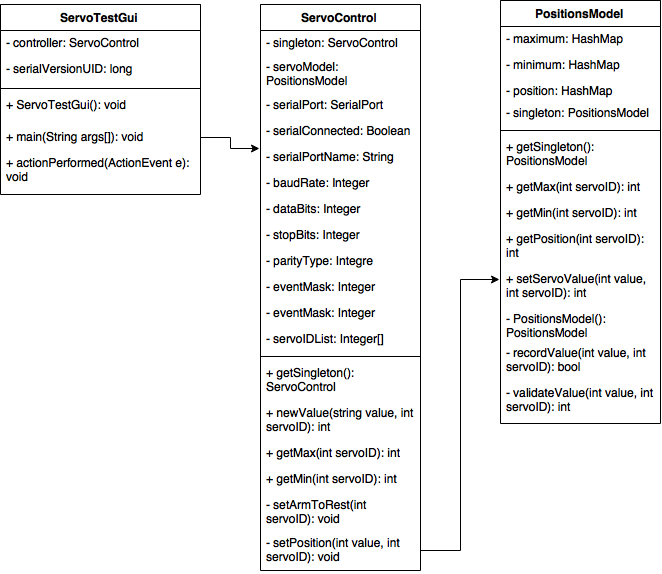


Figure #9 - User Story #796 Class Diagram

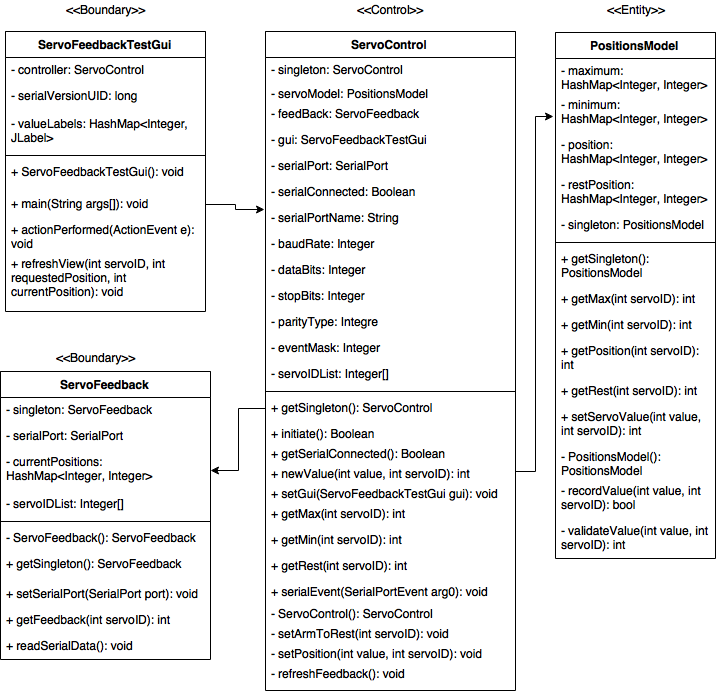


Figure #10 - User Story #808 Class Diagram

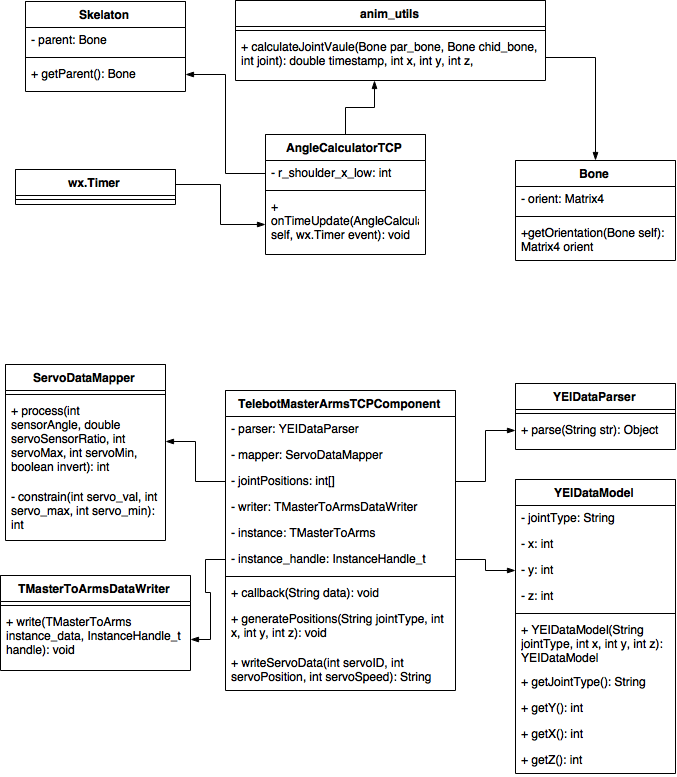


Figure #11 - User Story #816 Class Diagram

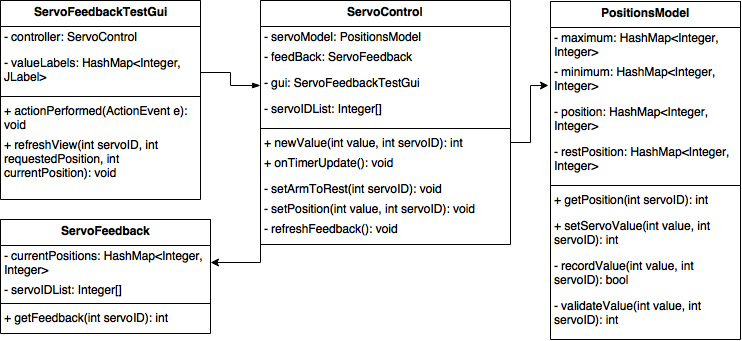


Figure #12 - User Story #839 Class Diagram

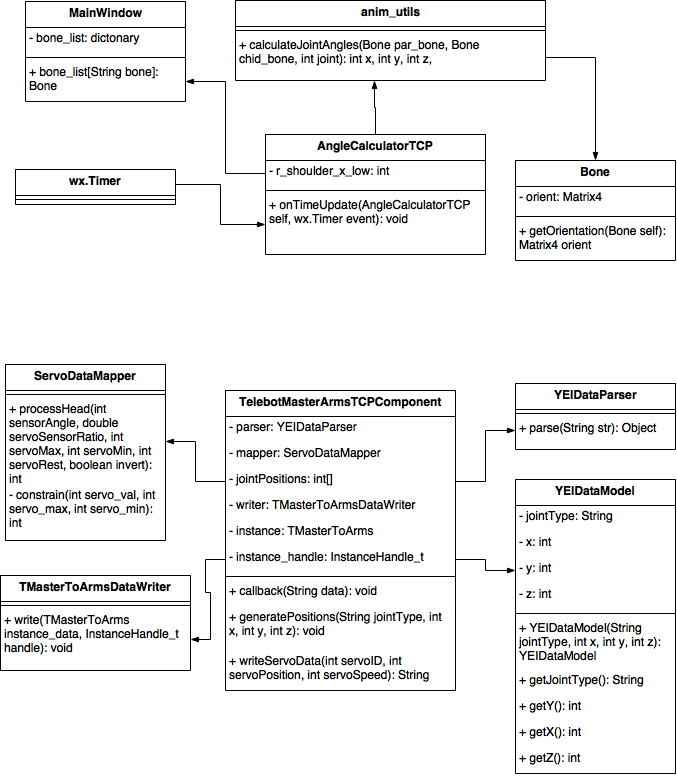


Figure #13 - User Story #840 Class Diagram

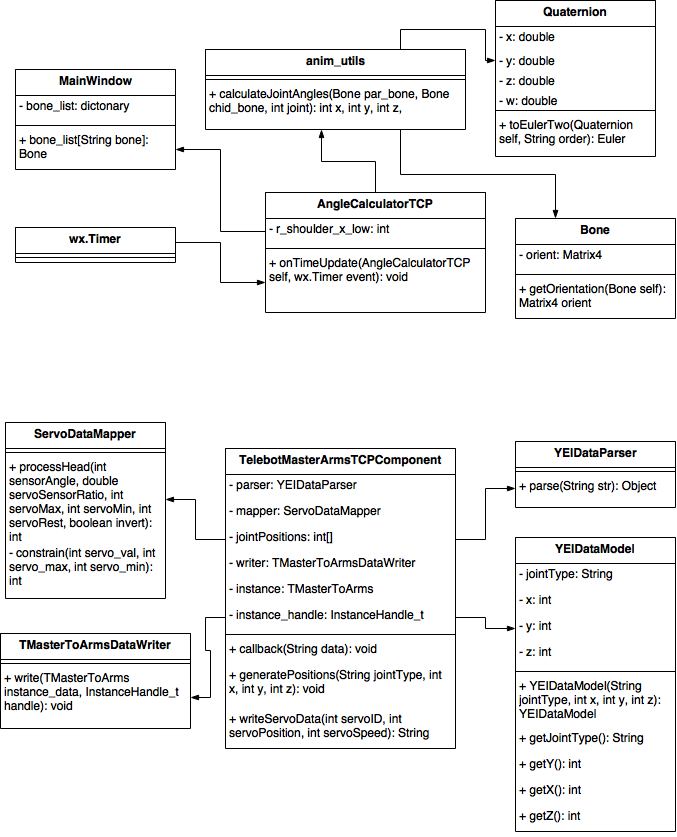


Figure #14 - User Story #851 Class Diagram

### Dynamic UML Diagrams

694UCD.png

Figure #15 - User Story #694 Use Case Diagram

694SD.png

Figure #16 - User Story #694 Sequence Diagram

735UCD.png

Figure #17 - User Story #735 Use Case Diagram

735SD.png

Figure #18 - User Story #735 Sequence Diagram

742UCD.png

Figure #19 - User Story #742 Use Case Diagram

742SD.png

Figure #20 - User Story #742 Sequence Diagram

743UCD.png

Figure #21 - User Story #743 Use Case Diagram

743SD.png

Figure #22 - User Story #743 Sequence Diagram

768UCD.png

Figure #23 - User Story #768 Use Case Diagram

768SD.png

Figure #24 - User Story #768 Sequence Diagram

769UCD.png

Figure #25 - User Story #769 Use Case Diagram

769SD.png

Figure #26 - User Story #769 Sequence Diagram

770UCD.png

Figure #27 - User Story #770 Use Case Diagram

770SD.png

Figure #28 - User Story #770 Sequence Diagram

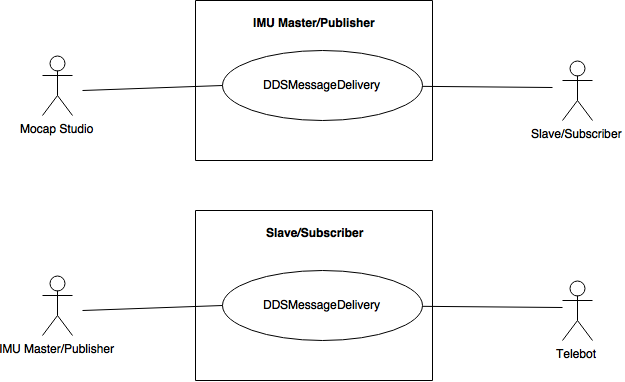


Figure #29 - User Story #789 Use Case Diagram

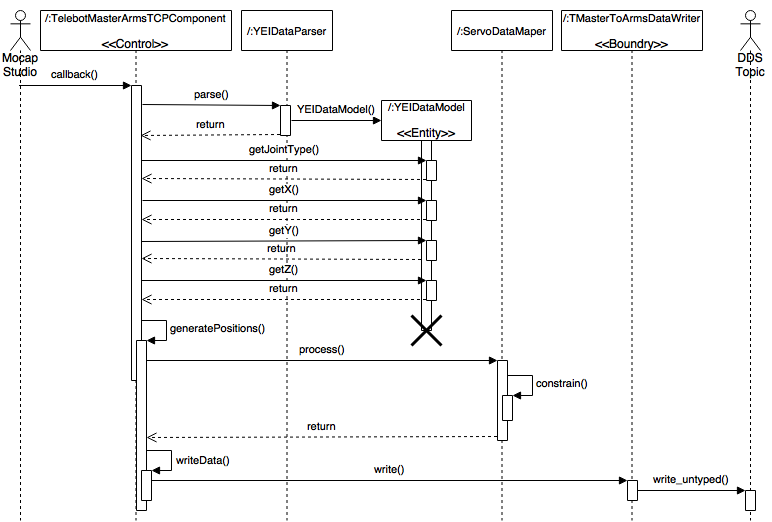


Figure #30 - User Story #789 Publisher Sequence Diagram

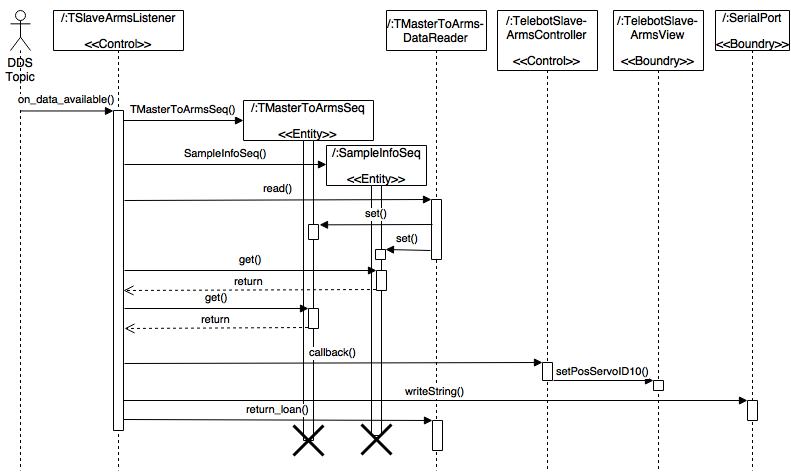


Figure #31 - User Story #789 Subscriber Sequence Diagram

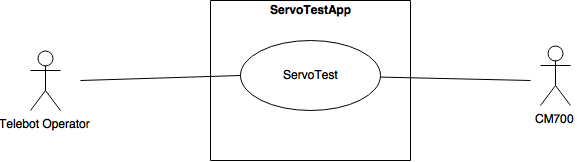


Figure #32 - User Story #796 Use Case Diagram

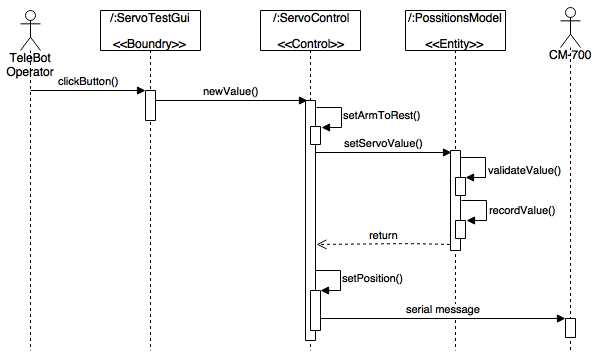


Figure #33 - User Story #796 Sequence Diagram

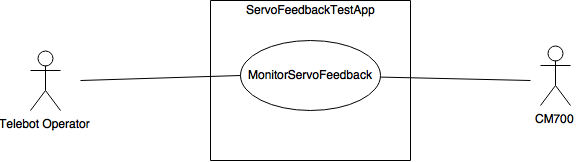


Figure #34 - User Story #808 Use Case Diagram

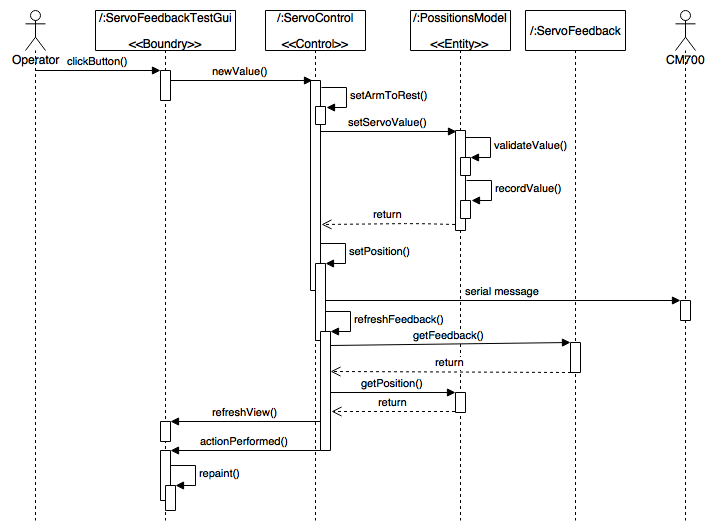


Figure #35 - User Story #808 Sequence Diagram

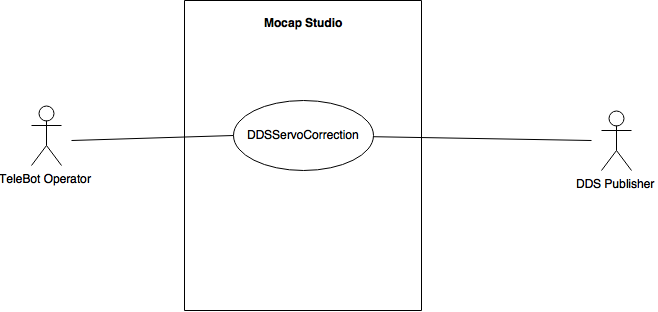


Figure #36 - User Story #816 Use Case Diagram

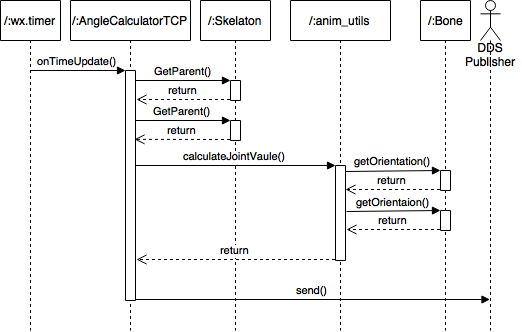


Figure #37 - User Story #816 Mocap Studio Sequence Diagram

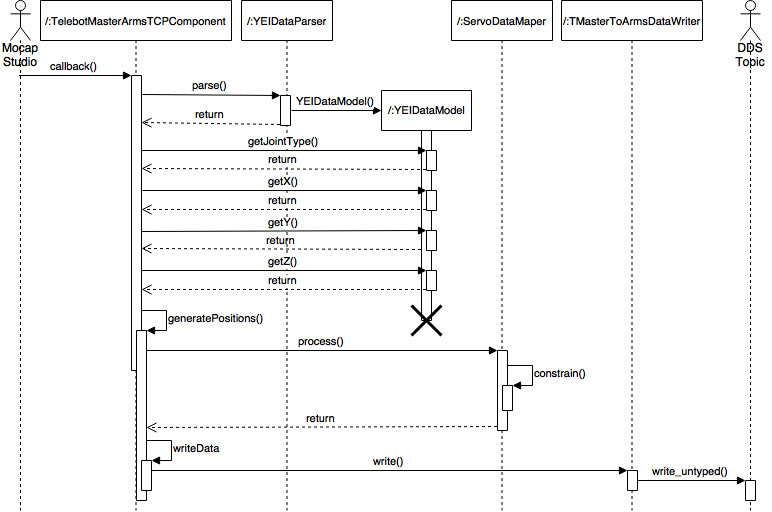


Figure #38 - User Story #816 IMU Master/Publisher Sequence Diagram

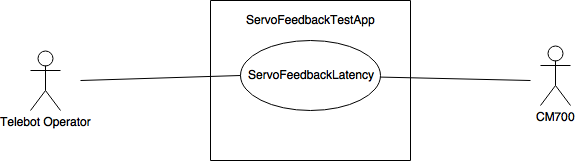


Figure #39 - User Story #839 Use Case Diagram

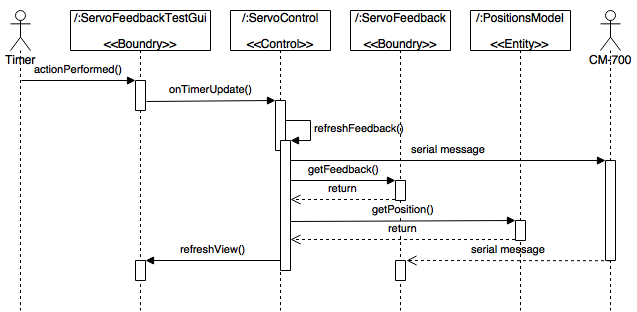


Figure #40 - User Story #839 Timer Sequence Diagram

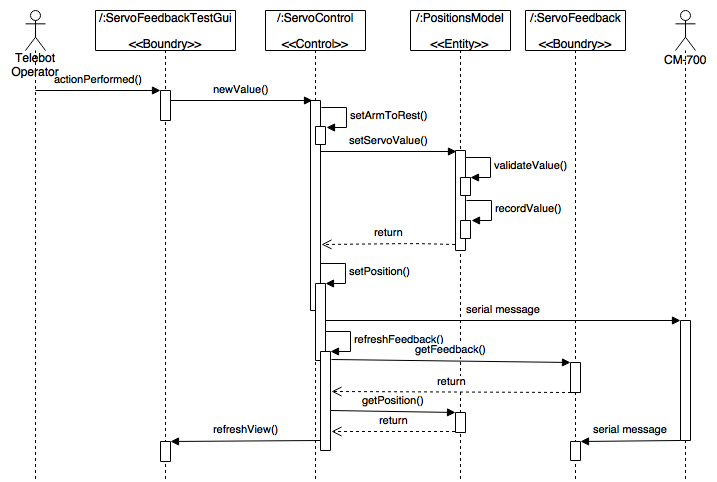


Figure #41 - User Story #839 Main Sequence Diagram

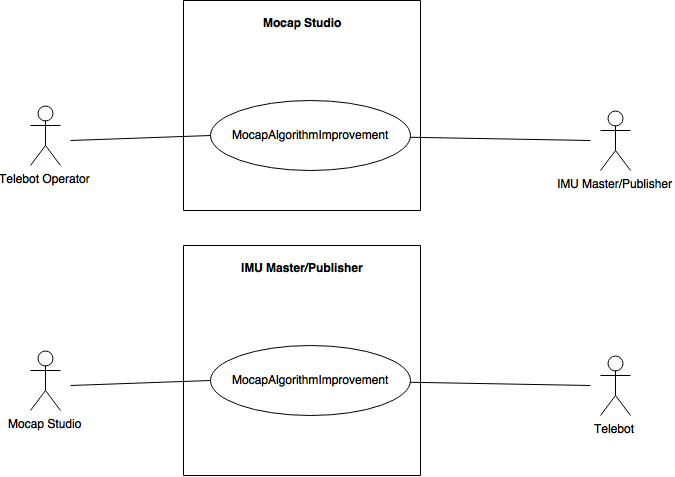


Figure #42 - User Story #840 Use Case Diagram

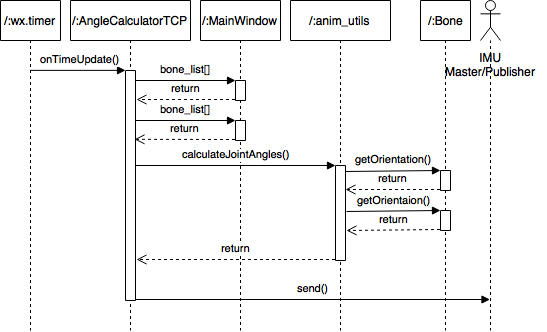


Figure #43 - User Story #840 Mocap Studio Sequence Diagram

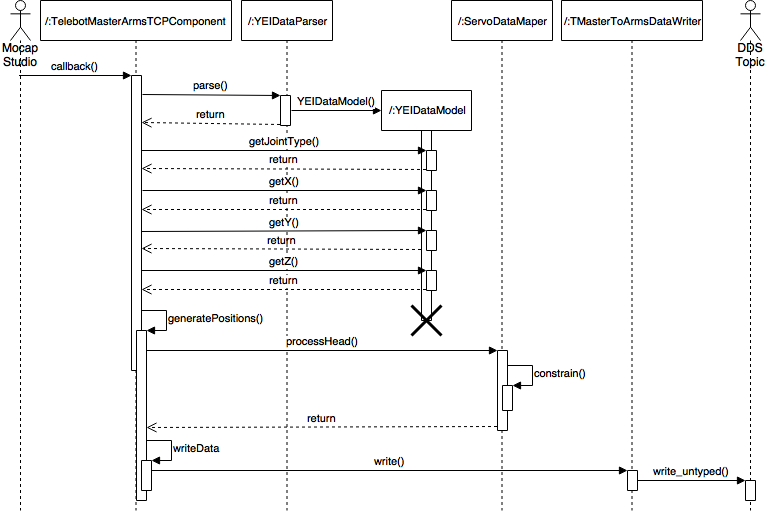


Figure #44 - User Story #840 IMU Master/Publisher Sequence Diagram

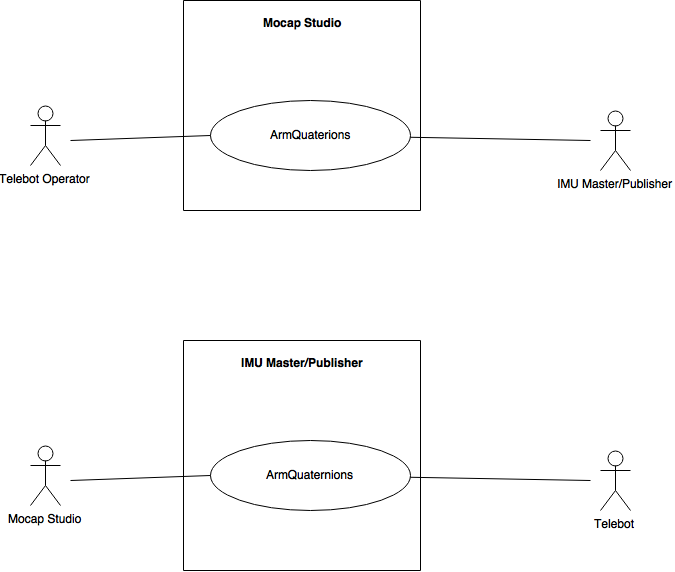


Figure #45 - User Story #851 Use Case Diagram

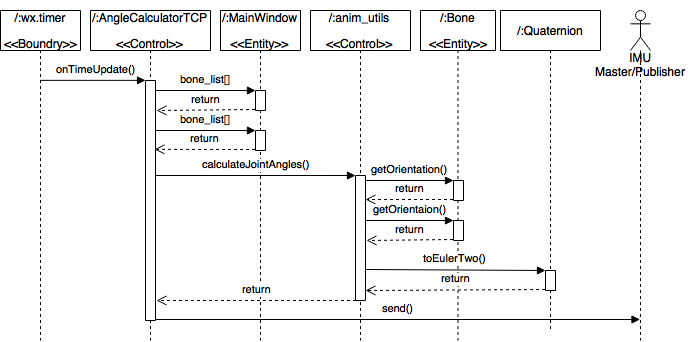


Figure #46 - User Story #851 Mocap Studio Sequence Diagram

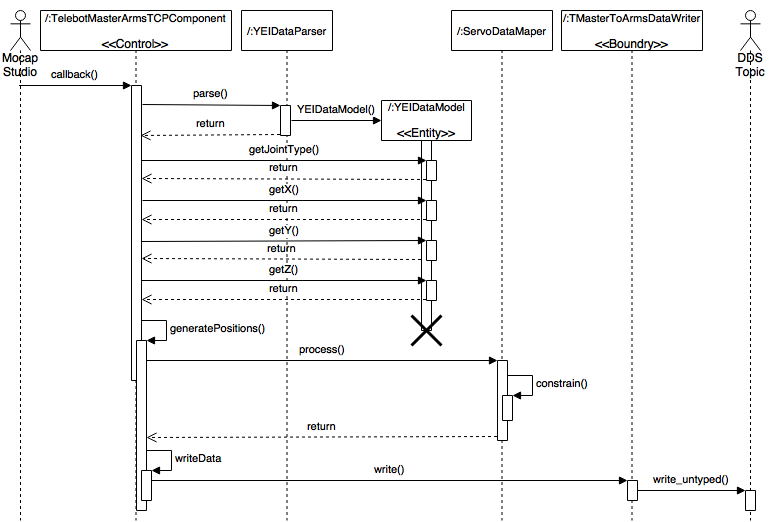


Figure #47 - User Story #851 IMU Master/Publisher Sequence Diagram

## Appendix B - User Interface Design

Include screenshots of the user interface of your system. For new versions of existing sytems, include only screenshots of the new or modified aspects of the user interface.

There’s no need for introducing this section.

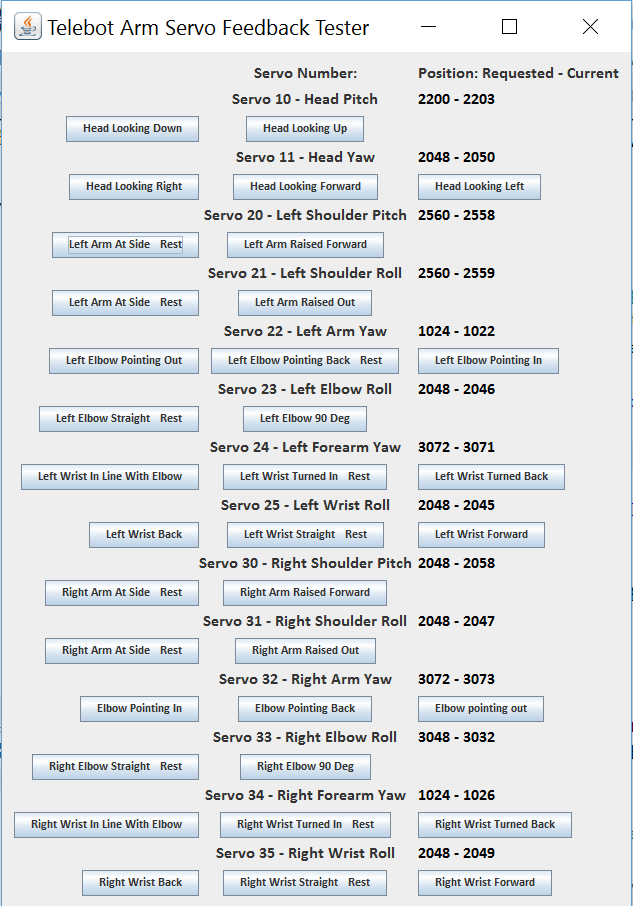


Figure # - Telebot Arm Servo Feedback Test App GUI

## Appendix C - Sprint Review Reports

**Sprint 1 Report**

**Date:** January 31, 2016

**Attendees:** Curtis Cox, Shadeh Ferris-Francis, Nagarajan Prabakar

**Topics Discussed:**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* None

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* User Story #789
  + Reason for rejection:
    - DDS Lag was not identified. Test suite creation was planned to help identify the problems
  + How this should be reflected on the user story definition in Mingle:
    - Test Suite will be created that can assisting in completing this story in the future.

**Sprint 2 Report**

**Date:** February 14, 2016

**Attendees:** Curtis Cox, Shadeh Ferris-Francis, Nagarajan Prabakar

**Topics Discussed:**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* None, due to infeasibility and incompleteness

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* User Story #800
  + Reason for rejection:
    - Infeasible, development and testing could not be completed due to errors in Mocap system on Windows VM
  + How this should be reflected on the user story definition in Mingle:
    - Moved to Defect Backlog
* User Story #796
  + Reason for rejection:
    - Integration testing incomplete.

## How this should be reflected on the user story definition in Mingle:

## Returned to backlog to be completed in Sprint 3

## Sprint 3 Report

## Date: February 28, 2016

## Attendees: Curtis Cox, Shadeh Ferris-Francis, Nagarajan Prabakar

## Topics Discussed:

## After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

## User Story #789

## User Story #808

## The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

## User Story #742

## Reason for rejection:

## Integration testing was not completed, as dummy data could not be sent to the IMU Master/Publisher without sensors present

## How this should be reflected on the user story definition in Mingle:

## Pushed to the next Sprint

## Sprint 4 Report

## Date: March 13, 2016

## Attendees: Curtis Cox, Shadeh Ferris-Francis, Nagarajan Prabakar

## Topics Discussed:

## After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

## User Story #816

## User Story #735

## User Story #742

## The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

## User Story #743

## Reason for rejection:

## Was not completed

## How this should be reflected on the user story definition in Mingle:

## To be finished in Sprint 5

## Sprint 5 Report

## Date: April 3, 2016

## Attendees: Curtis Cox, Shadeh Ferris-Francis, Nagarajan Prabakar

## Topics Discussed:

## After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

## User Story #839

## User Story #840

## User Story #694

## User Story #743

## The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

## None

## Sprint 6 Report

## Date: April 14, 2016

## Attendees: Curtis Cox, Shadeh Ferris-Francis, Nagarajan Prabakar

## Topics Discussed:

## After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

## User Story #851

## User Story #770

## The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

## User Story # 789

## Reason for rejection:

## Not completed

## How this should be reflected on the user story definition in Mingle:

## Return to backlog

## User Story # 807

## Reason for rejection:

## Not completed

## How this should be reflected on the user story definition in Mingle:

## Return to backlog

## User Story # 768

## Reason for rejection:

## Errors in program later discovered

## How this should be reflected on the user story definition in Mingle:

## Return to backlog

## User Story # 769

## Reason for rejection:

## Not completed

## How this should be reflected on the user story definition in Mingle:

## Return to backlog

## Sprint 7 Report

## Date: April 28, 2016

## Attendees: Curtis Cox, Shadeh Ferris-Francis, Nagarajan Prabakar

## Topics Discussed:

## After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

## User Story #768

## User Story #769

## User Story #789

## The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

## None

## 

## Appendix D - Sprint Retrospective Reports

**Sprint 1 Retrospective**

**Date:** January 31, 2016

**Attendees:** Curtis Cox, Shadeh Ferris-Francis

**Discussed Topics:**

What went wrong?

* Did we do a good job estimating our team's velocity?

No, we did not estimate our velocity.

* Did we do a good job estimating the points (time required) for each user story?

We underestimated the time required for testing.

* Did each team member work as scheduled?

Student 1 did with the exception of Friday evening January 29, 2016.

Student 2 needs to allocate a regular timeframe for the following sprints.

How to address the issues in the next sprint?

* How to improve the process?
  + Allocating more time

**Sprint 2 Retrospective**

**Date:** February 14, 2016

**Attendees:** Curtis Cox, Shadeh Ferris-Francis

**Discussed Topics:**

What went wrong?

* Did we do a good job estimating our team's velocity?

No, due to unforeseen issues and troubleshooting

* Did we do a good job estimating the points (time required) for each user story?

No; troubleshooting (hardware and software) altered estimated time

* Did each team member work as scheduled?

Student 1 : My current work schedule is very fluid. I have had to improvise my schedule hour to hour, but I have tried to reflect the actual number of hours worked and stayed to the schedule as much as work would allow.

Student 2 : More hours required to fulfill 20 hour requirement

What went right?

* Hardware wise, robot is now responsive due to re-installation of firmware

How to address the issues in the next sprint?

* How to improve the process?
  + A given time to abandon a user story which seems infeasible should be established.
* How to improve the product?
  + Packaging all firmware and dependencies

**Sprint 3 Retrospective**

**Date:** February 28, 2016

**Attendees:** Curtis Cox, Shadeh Ferris-Francis

**Discussed Topics:**

What went wrong?

* Did we do a good job estimating our team's velocity?

We overestimated our ability to finish stories.

* Did we do a good job estimating the points (time required) for each user story?

We underestimated the amount of time that many individual tasks would take.

* Did each team member work as scheduled?

Student 1 : During the first week of the sprint I only worked about half of my scheduled hours. I worked according to my schedule, plus make-up hours during the second week.

Student 2 : More time was allocated, but need to get 20 hours each week

What went right?

* Hardware wise, robot is now responsive due to re-installation of firmware

How to address the issues in the next sprint?

* How to improve the process?
  + More communication between product owners (incl. lab members) and project members so less conflict

**Sprint 4 Retrospective**

**Date:** March 13, 2016

**Attendees:** Curtis Cox, Shadeh Ferris-Francis

**Discussed Topics:**

What went wrong?

* Did we do a good job estimating our team's velocity?

We completed 3 out of 4 user stories, which was an improvement

* Did we do a good job estimating the points (time required) for each user story?

No, tasks that should have taken less time consumed more time than stated

* Did each team member work as scheduled?

Student 1 : No, need to try harder to stick to scheduled hours.

Student 2 : No, need to fulfill 20 hour requirement

What went right?

* Completion of user stories and being able to collaborate/give feedback when needed

How to address the issues in the next sprint?

* How to improve the process?
  + Get more frequent feedback from Product owner
* How to improve the product?
  + Dynamic calibration of sensors

**Sprint 5 Retrospective**

**Date:** April 3, 2016

**Attendees:** Curtis Cox, Shadeh Ferris-Francis

**Discussed Topics:**

What went wrong?

* Did we do a good job estimating our team's velocity?

We completed 4 out of 4 user stories, which was our velocity

* Did we do a good job estimating the points (time required) for each user story?

No, documentation took longer than time required (Student 2)

No, I ran into bugs that took longer than they should have to remove (student 1)

* Did each team member work as scheduled?

Student 1 : Yes, for the most part

Student 2 : No, need to fulfill 20 hour requirement

What went right?

* Completion of user stories and being able to collaborate/give feedback when needed

How to address the issues in the next sprint?

* How to improve the process?
  + N/A
* How to improve the product?
  + Less latency (lag in robot)

**Sprint 6 Retrospective**

**Date:** April 15, 2016

**Attendees:** Curtis Cox, Shadeh Ferris-Francis

**Discussed Topics:**

What went wrong?

* Did we do a good job estimating our team's velocity?

We completed 2 out of 4 user stories

* Did we do a good job estimating the points (time required) for each user story?

No, I underestimated the amount of time that would be needed to solve the equations for quaternion rotations that needed to be coded. (student 1)

* Did each team member work as scheduled?

Student 1 : No, I missed my evening work time five times

Student 2 : No, need to fulfill 20 hour requirement

What went right?

* We kept working on the problems that we ran into until the solution was found.

How to address the issues in the next sprint?

* How to improve the process?
  + N/A
* How to improve the product?
  + Less latency (lag in robot)

**Sprint 7 Retrospective**

**Date:** April 28, 2016

**Attendees:** Curtis Cox, Shadeh Ferris-Francis

**Discussed Topics:**

What went wrong?

* Did we do a good job estimating our team's velocity?

We completed 3 out of 3 user stories

* Did we do a good job estimating the points (time required) for each user story?

Relatively good job

* Did each team member work as scheduled?

Student 1 : No, The first week I did not get 20 hours but made up for it in the second week

Student 2: The last week of Sprint, was able to fulfill requirement with more additional hours

What went right?

* We worked well as a team to accomplish the things that we could not do alone.

# 

# 

# References

Mocap Studio is a product of Yost Labs <https://www.yostlabs.com/yei-3-space-mocap-studio>

Connext DDS is a product of RTI <http://www.rti.com/>

For User Story #851 Arm Quaternions:

toEulerTwo() converts a Quaterion to xyx, xzx or yzy euler angles.

It uses an Body fixed rotation version of the world fixed rotation

Euler Angle Formulas published at

http://www.geometrictools.com/Documentation/EulerAngles.pdf

Euler Angle Formulas

David Eberly

Geometric Tools, LLC

http://www.geometrictools.com/

Copyright

c 1998-2016. All Rights Reserved.

Created: December 1, 1999

Last Modied: March 10, 2014

Body Fixed Rotation Matrices from:

http://www.control.aau.dk/~jan/undervisning/MechanicsI/mm2pres.pdf Page 12